NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UNCOOLED INFRARED IMAGING FACE RECOGNITION USING KERNEL-BASED FEATURE VECTOR SELECTION

by

Ioannis M. Alexandropoulos

September 2006

Thesis Committee Supervisor:	Monique P. Fargues
Thesis Committee Members:	Roberto Cristi
	Carlos Borges

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2006	**3. REPORT TYPE AND DATES COVERED** Engineer's Thesis	
4. TITLE AND SUBTITLE: Uncooled Infrared Imaging Face Recognition Using Kernel-Based Feature Vector Selection			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ioannis M. Alexandropoulos			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

A considerable amount of research has been recently conducted on face recognition tasks, due to increasing demands for security and authentication applications. Recent technological developments in uncooled IR imagery technology have boosted IR face recognition research applications. Our study is part of an on-going research initiated at the Naval Postgraduate School that considers an uncooled low-resolution and low-cost IR camera used for face recognition applications. This work investigates a recent approach which approximates nonlinear kernel-based methods at a significantly reduced computational cost. Our research was applied to an IR database. Results show that this scheme may perform sufficiently close to its "kernelized" version considered in a previous study, at a fraction of the computational cost, provided that the associated parameters are well tuned. The thesis considers a relative comparison between the two algorithms, based on identification and verification experiments and considers a statistical test to investigate whether classification performance differences may be considered statistically significant. Results show that, from a cost perspective, a low-resolution uncooled IR camera in conjunction with a low computational-cost classification scheme can be embedded in a robust face recognition system to efficiently address the issue of authentication in security-related tasks.

14. SUBJECT TERMS Face Recognition, Pattern Classification, Infrared, FVS, Distances, Feature Vectors			**15. NUMBER OF PAGES** 155
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UNCOOLED INFRARED IMAGING FACE RECOGNITION USING KERNEL-BASED FEATURE VECTOR SELECTION

Ioannis M. Alexandropoulos
Lieutenant Junior Grade, Hellenic Navy
B.S., Hellenic Naval Academy, 1999

Submitted in partial fulfillment of the
requirements for the degrees of

ELECTRICAL ENGINEER
and
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2006

Author: Ioannis M. Alexandropoulos

Approved by: Monique P. Fargues
Thesis Committee Supervisor

Roberto Cristi
Thesis Committee Member

Carlos Borges
Thesis Committee Member

Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A considerable amount of research has been recently conducted on face recognition tasks, due to increasing demands for security and authentication applications. Recent technological developments in uncooled IR imagery technology have boosted IR face recognition research applications. Our study is part of an on-going research initiated at the Naval Postgraduate School that considers an uncooled low-resolution and low-cost IR camera used for face recognition applications. This work investigates a recent approach which approximates nonlinear kernel-based methods at a significantly reduced computational cost. Our research was applied to an IR database. Results show that this scheme may perform sufficiently close to its "kernelized" version considered in a previous study, at a fraction of the computational cost, provided that the associated parameters are well tuned. The thesis considers a relative comparison between the two algorithms, based on identification and verifications experiments and considers a statistical test to investigate whether classification performance differences may be considered statistically significant. Results show that, from a cost perspective, a low-resolution uncooled IR camera in conjunction with a low computational-cost classification scheme can be embedded in a robust face recognition system to efficiently address the issue of authentication in security-related tasks.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

To my parents, Michail and Paraskevi and my wife Maria, for her enduring support.

Also to my thesis advisor Dr. Monique P. Fargues, for her guidance, professionalism, encouragement and patience that made the subject study a worthwhile learning experience. Finally I am grateful to Dr Fatiha Anouar and Gaston Baudat for their willingness in sharing their Matlab code for Feature Vector Selection scheme and providing additional information regarding this scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

A considerable amount of research has been conducted on face recognition tasks over the last few years, due primarily to the rapidly increasing demand for alternative means to ensure security and authentication. Traditional means of identification such as ID cards and passwords are vulnerable to being compromised, unlike face recognition, which may offer a robust natural mean of identification. In the past, most of the research was focused on visible imaging, due to the high cost of infrared (IR) cooled cameras in conjunction with low-resolution image analysis. Recently, however, IR camera technology has significantly improved, resulting in improvements also in uncooled IR camera sensitivity (i.e., higher resolution) at a fraction of the cost associated with cooled devices.

This study is an extension of the research conducted by [Domboulas, 2004], which investigated a nonlinear kernel-based classification scheme, the Generalized Discriminant Analysis (GDA). The GDA scheme was applied to an IR database, first generated by Pereira [Pereira, 2002] to include fourteen adult subjects collected in a controlled indoor environment using an uncooled low-resolution IR camera (IR-160). The database was further expanded by Lee to a total of fifty adult subjects [Lee, 2004]. Both Pereira's and Lee's studies considered two classic linear classification schemes, namely, the Principal Components Analysis (PCA) and the Linear Discriminant Analysis (LDA).

In our work, we first considered a linear scheme implementing the Bayes Classification algorithm. Next, a novel approach to nonlinear kernel methods was implemented by using a data selection process called the "Features Vector Selection" (FVS). Use of the FVS scheme followed by the classic linear classification scheme LDA can achieve performances similar to those obtained with the GDA method at a significantly reduced computational cost, since only a portion of the available data is used for extracting features that best represent the dataset.

We considered two different types of experiments in order to evaluate the performance of the FVS method followed by the classic linear classification LDA scheme. Specifically, we investigated identification and verification experiments along with their performance metrics and Cumulative Match Characteristic (CMC) and Receiver Operating Characteristic (ROC) curves. Kernel-functions tuning issues were discussed, while parameters for the Gaussian and the polynomial kernels in conjunction with three different types of distances (i.e., the Euclidean norm, the Euclidean angle, and the Mahalanobis angle) were selected, using the 5-fold cross-validation scheme. The thesis also investigates whether differences in the performance of the two schemes, may be considered as statistically significant in the identification experiment scenario. Results show that selecting the number of features for the FVS-LDA scheme from a third to a half of the available features results in close to the standard GDA performance at a fraction of the computational cost.

Our research concludes that, from a cost perspective, a low-resolution uncooled IR camera in conjunction with a low computational-cost classification scheme can be embedded in a robust face recognition system to efficiently address the issue of authentication in security-related tasks.

I. INTRODUCTION

In the last few years, a considerable amount of research has been conducted on face recognition tasks [Chen, 2005], [Socolinsky, 2003], [Lu, 2003], [Wang, 2004], [Thomaz, 2005]. It was mostly based on the rapidly increasing demand for alternative means for security and authentication. Traditional means of identification such as ID cards and passwords are vulnerable to compromise, unlike face recognition, which offers "a non-intrusive, and probably the most natural way of identification" [Kong, 2005]. Although most of the face recognition research is based on visible imaging, visual face recognition-based systems perform poorly under poor illumination conditions and in distinguishing skin-color variations [Prokoski, 2000]. An alternative approach for illumination invariant face recognition tasks is the thermal infrared (IR) imagery. However, although it is a promising alternative, at first it received little attention, due to the high cost of IR cameras in conjunction with low-resolution image analysis. Recently, IR camera technology has been significantly improved, which led to improvements also in IR camera sensitivity (i.e., higher resolution) and price reductions. Indeed, it is those factors that boosted the IR face recognition research [Socolinsky, 2001], [Chen, 2003].

This study is an extension of the research conducted by [Domboulas, 2004], which investigated a nonlinear kernel-based classification scheme, the Generalized Discriminant Analysis (GDA) proposed by Baudat and Anouar [Baudat, 2000]. The GDA scheme was applied to an IR database, first generated by Pereira [Pereira, 2002] to include fourteen adult subjects collected in a controlled indoor environment, and was further expanded by Lee to a total of fifty adult subjects [Lee, 2004]. Both Pereira's and Lee's studies considered two classic linear classification schemes, namely, the Principal Components Analysis (PCA) and the Linear Discriminant Analysis (LDA). This study investigated in our work first considers a linear scheme implementing the Bayes Classification algorithm. Next, a recently proposed approach to nonlinear kernel methods is implemented by using a data selection process called "Features Vector Selection" (FVS) [Baudat, 2003]. Use of the FVS scheme followed by the classic linear classification scheme LDA can achieve performances similar to the GDA method at a significantly reduced computational cost. In fact, FVS-LDA turns out to be a good

approximation of its "kernelized" version (i.e., GDA) [Baudat, 2003]. This study also considers the relative classification performance of the FVS-LDA and GDA methods.

A. THEORETICAL BACKGROUND

One type of electromagnetic radiation that has received a lot of attention recently is infrared (IR) radiation. IR radiation refers to the region beyond the red end of the visible color spectrum, the region located between the visible region and the microwave region of the electromagnetic spectrum [FLIR, 2006].

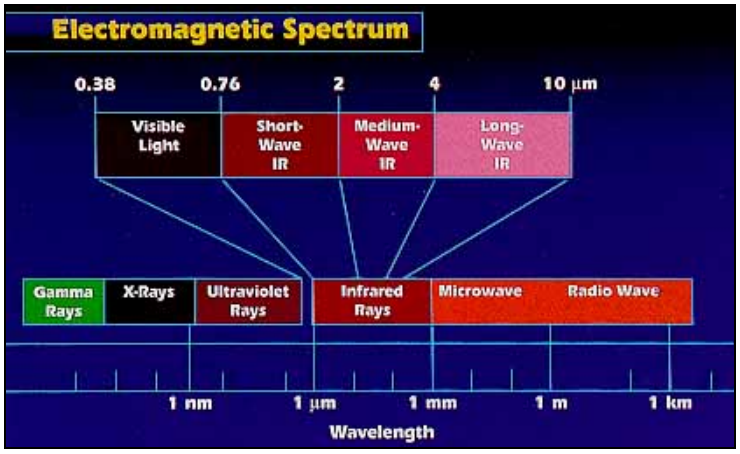

Figure 1.1 Electromagnetic spectrum wavelengths (from [Berkeley, 2006]).

A crucial feature of IR applications in imaging technology is that, by using IR sensors, we are able to capture information that is not visible. Note that all objects radiate some energy in the infrared, even objects at room temperature and frozen objects such as ice [FLIR, 2006]. Thus, even though IR wavelengths are not in the visible spectrum, IR radiation can be measured in terms of the thermal energy radiated.

Next, we briefly introduce some of the basic principles behind IR applications. First, as infrared energy strikes an object, it can be

1. Abstracted and Emitted (E),
2. Reflected (R), and
3. Transmitted (T).

Because most materials do not transmit infrared energy, for a total amount of energy E_{IR} the following is true: $E_{IR}=E+R$. The measure of materials' features to absorb and emit or to reflect IR energy are the *emissivity E* and the *reflectance R*, respectively. Emissivity and reflectance values are bounded between zero and one; in practice, we usually refer mostly to a material's emissivity values. Emissivity is defined as the ratio of the IR energy that an object absorbs and radiates with respect to the energy that a "black body" (i.e., perfect emmiter, emissivity one) would radiate. Materials of high emissivity emit IR more than they reflect it; thus good reflectors appear to be of the same temperature as their background and are not distinguishable by an IR sensor. The amount of radiation emitted is proportional to the temperature of the object and its emissivity. Next, we present the human-face characteristics that are well suited for IR image face recognition applications.

The scope of the research involving biometrics that is conducted by the security industry is aimed at developing identification schemes that are based on characteristics unique to a single individual (e.g., fingerprints). According to Prokoski, "the anatomical information which is utilized by infrared identification involves subsurface features unique to each person" [Prokoski, 2000]. Interestingly, even identical twins appear to have different thermal face patterns. The mechanism behind the collection of IR images is related to the pattern of superficial blood vessels that transfer heat throughout the human body. The area of the skin that is directly above a blood vessel is, on average, $0.1\,^{o}C$ warmer than the adjacent skin [Prokoski, 2000]. Moreover, the temperature variation for a typical human face is in the range of about $8\,^{o}C$ [Prokoski, 2000]. Recent improvements in IR-imaging technology enable us to capture details for the values of temperature range and sensitivity.

A complementary approach to IR imaging, namely, visual imaging based on the reflectance information of a face object, has received most of the recent research attention, due to the low cost and high resolution of visible-light spectrum cameras. However, changes in illumination introduce variability in the within-class information that results in the degradation of a classification's performance. In fact, variations between the images of the same face due to changes in illumination, viewing direction,

facial expressions, and pose are typically larger than the variations introduced when different objects are considered [Kong, 2005]. Thermal IR imagery is invariant to those types of distortions, since it captures the anatomical information. However, thermal imaging has limitations in identifying a person wearing glasses, since glass is a material of low emissivity, or when the thermal characteristics of a face have changed due to increased body temperature (e.g., physical exercise) [Kong, 2005].

B. THE DATABASE CONSIDERED AND THE EQUIPMENT USED

The IR-images database considered in this study was initially collected by Pereira [Pereira, 2002] and further extended by Lee [Lee, 2004] at the Naval Postgraduate School (NPS). The IR camera used (IR-160) was developed by Infrared Solutions Inc. and belongs to the family of uncooled (i.e., no cryogenic cooling of the sensor is required) IR long-wavelength thermal imagers. The IR-160 imager provides a $(160 \times 120$ pixel) NTSC or PAL video signal, it is tuned to the long-wavelength infrared band (LWIR, 8–14 μm), and its sensitivity is about 60mK (i.e., less than $0.1\,^{o}C$) [Infrared Solutions Inc., 2006].

Although the performance characteristics of the IR-160 camera are relatively poor, the operational demands related to face imaging are well within the spatial and temperature resolutions provided by the camera. More specifically, recall that the temperature variation for a typical human face is in the range of about $8\,^{o}C$ and that the area of the skin that is directly above a blood vessel is on average $0.1\,^{o}C$ warmer than the adjacent skin [Prokoski, 2000]. Moreover, "humans, at normal body temperature, radiate most strongly in the infrared at a wavelength of about 10 μm" [NASA, 2006]. The blackbody spectrum for temperatures near that of the human body is depicted in Figure 1.2.

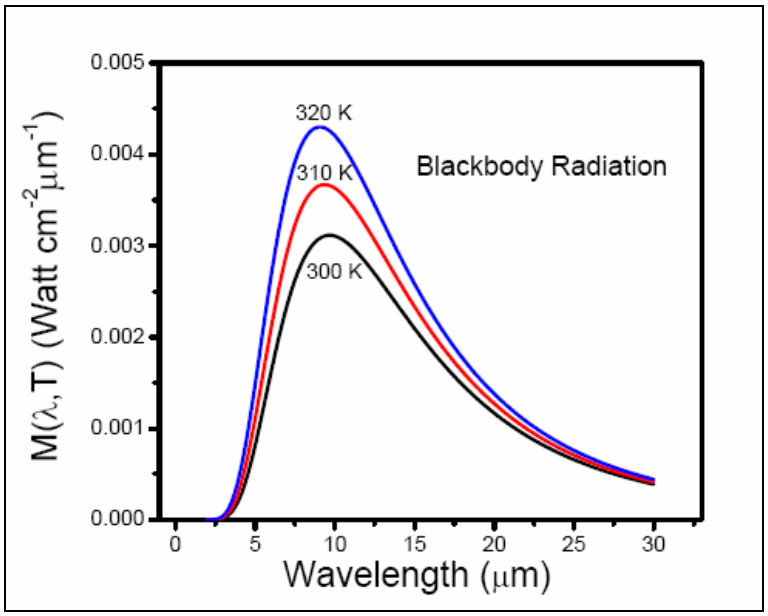

Figure 1.2 Blackbody spectrum for temperatures near that of the human body (from [Pereira, 2002]).

The IR-160 imager is connected via its RS-232 serial communication link to a monitor, thus allowing for a manual optimization of the images (e.g., brightness and contrast adjustments) before their collection. Images are 8-bit/pixel with 160×120 pixels spatial and 60mK temperature resolutions. The database includes frontal views of 50 adult subjects, each sampled 10 times with 3 facial expressions each, resulting in 1,500 grey-scale images. Images were cropped down to 60×45 pixels to retain the middle section of the face only [Lee, 2004].

Specifically, the variations introduced in terms of facial expressions included neutral, smiling, and pronouncing the vowel "u" expressions. Further variations were introduced for a given facial expression by considering nine fixed, predefined gazing directions for the individuals, while a tenth image variation was collected for the random-gazing positions. Note that variation due to the time-lapse between images of the same individual collected at different time periods was not considered. Additional information regarding the database collection and the IR images' final formation can be found in [Lee, 2004].

The equipment used (i.e., imager IR-160, monitor, and PC) along with the lateral and frontal aspect layouts set-up followed for the IR data collection system are depicted in the following figures.

Figure 1.3 Equipment used for IR-images database collection (from [Lee, 2004]).

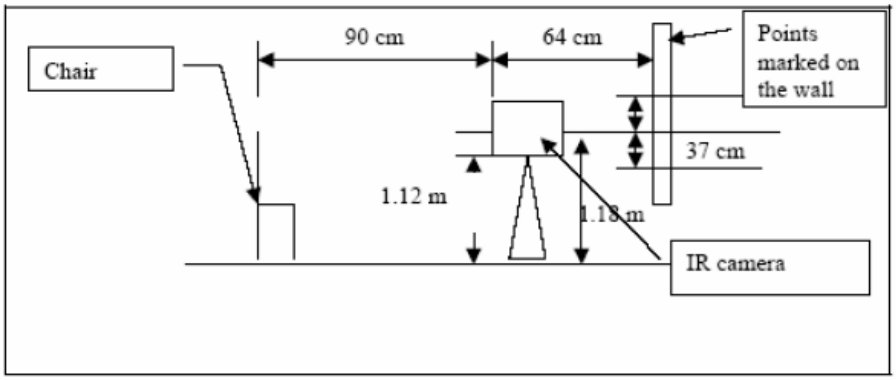

Figure 1.4 Lateral aspect of IR data-collection system layout (from [Pereira, 2002]).

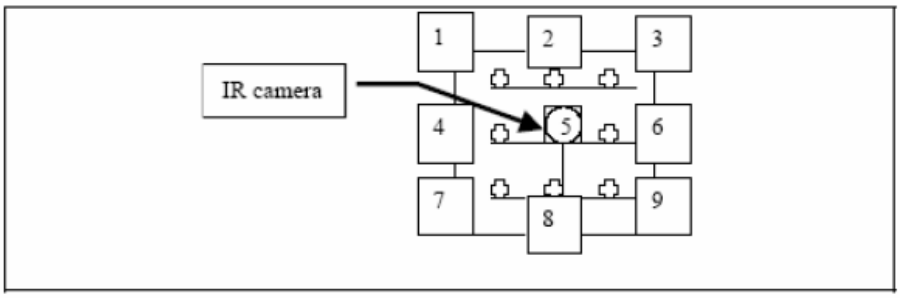

Figure 1.5 Forward aspect of IR data-collection system layout (from [Pereira, 2002]).

C. THESIS OVERVIEW

This study consists of five chapters. Chapter I introduced the idea behind IR imaging face recognition tasks. Chapter II presents the underlying theory of linear classification schemes. Nonlinear classifiers and specifically, a generalization of linear schemes using nonlinear kernel functions are discussed in Chapter III. Experimental results obtained using the IR-images database are illustrated and discussed in Chapter IV. Finally, Chapter V includes conclusions and proposals for future work. Appendix A contains mathematical background information when more details are needed. The software implementation considered for this study is included in Appendix B.

D. SUMMARY

This chapter briefly introduced some basic concepts related to IR imaging applications. Specifically, we discussed the mechanism behind IR-imaging face recognition tasks. A comparison with an alternative (i.e., visual imaging) approach was considered and major shortcomings and limitations noted. Next we presented the process followed for the IR-images database collection along with the equipment used. Next section, we first consider two classical linear-classification schemes, and second, review a nonlinear kernel-based scheme already investigated in a previous study [Domboulas, 2004].

THIS PAGE INTENTIONALLY LEFT BLANK

II. LINEAR CLASSIFICATION ALGORITHMS

This chapter first introduces three linear classification algorithms widely used in pattern recognition applications, namely the Principal Component Analysis (PCA), the Linear Discriminant Analysis (LDA), and the classifier based on Bayes Decision Theory. Next, it discusses similarities and deficiencies present in each.

A. INTRODUCTION

An important preprocessing step of an overall automatic pattern recognition system is feature extraction which is designed to capture relevant information with high discriminating capability. Facial recognition applications need to cope with large dimensionality issues, which are due to the nature of the imaging data. Note that, for the IR camera used in our study [Lee, 2004], images of size (160×120) collected were reshaped as column vectors of dimension $N = 19200,$ resulting in a very large covariance matrix of dimension $N \times N$. Decomposing matrices of such a dimension is computationally intensive unless dimensional reduction preprocessing schemes are applied. The PCA method, which was initially designed for data compression applications, is also applied in numerous pattern recognition tasks. However, a projection scheme designed to best represent the data in a least-square sense is not optimum for discrimination tasks, unlike the LDA algorithm that seeks a projection that best separates the data. In the Bayesian Classifier approach, PCA is first applied for dimensionality reduction, followed by a recognition step in the reduced subspace which follows the Maximum A-Posteriori (MAP) decision rule. Recall the MAP decision rule is optimal for classification since it considers the class separation. We consider each of these approaches applied to our IR image database and compare their performances.

B. PRINCIPAL COMPONENTS ANALYSIS

1. Introduction

Sirovic and Kirby are among the first researchers that introduced PCA for image face representation [Liu, 1998]. This approach leads to decomposing any image into its eigenpictures so that the image can later be economically reconstructed by selecting only a portion of the eigenpictures and the associated projections onto the eigenpictures subspace. Next, Turk and Pentland introduced the eigenface approach for face-

recognition applications in which eigenvectors associated to the larger eigenvalues (referred to as eigenfaces) of the image face covariance matrix decomposition span a feature subspace of smaller dimension [Liu, 1998]. Over the years PCA has been used in many pattern classification tasks, mostly due to its generalization ability, since the features (i.e., projection axes) extracted, which are based on the variations of all the training samples, are fairly robust for representing labeled images. However, the inefficiency of directly applying the PCA method is due to the fact that the null subspace information associated to the smaller eigenvalues is ignored, even though it contains useful class discrimination information [Lu, 2003]. In addition, PCA leads to poor performance when the distributions of face classes are not separated by the mean-differences but by the covariance-difference. In such a case, the most expressive features are not necessarily the most discriminating ones when the training images are representative of the range of class variations (i.e., between-class scatter information).

a. Algorithm Description

The PCA algorithm for pattern recognition is implemented in two steps. First, features from high-dimensional data are extracted. Next, a classifier scheme is designed on the training data and used to assign labels to the testing data.

b. Feature Extraction

Linear methods project the high-dimensional data onto a feature subspace of significantly lower dimension, and PCA is a powerful method for extracting features from high-dimensional data sets. That approach reduces the dimensionality of the feature space by first selecting the directions along which the data scatter is maximized [Duda, 2001]. Next, principal components are obtained by solving an eigenvalue problem. Assume we have *P* available images stored in a $m \times n$ matrix structure. First, each image is reshaped into a column vector structure $a_i \in \mathbb{R}^N, i = 1,...,P$ of dimension $N = mn$. Next, let us review the standard PCA algorithm steps, as illustrated in [Schölkopf and Smola, 2002]. According to Duda "the sample mean is a zero-dimensional representation of the data set. It is simple, but it does not reveal any variability in the data." Hence the given set of observations (i.e., images) is first centered, by subtracting the overall image mean vector $m \in \mathbb{R}^N$ defined as:

$$m = \frac{1}{P}\sum_{i=1}^{P} a_i \ . \tag{2.1}$$

Thus, the centered dataset is defined as:

$$x_i = a_i - m, i = 1,...,P \ \ x_i = a_i - m, i = 1,...,P, \tag{2.2}$$

resulting in the covariance matrix:

$$C = \frac{1}{P}\sum_{j=1}^{P} x_j x_j^T = \frac{1}{P} XX^T, C \in \mathbb{R}^{N \times N}, \tag{2.3}$$

where the data matrix X is defined as:

$$X = \left[x_1, x_2, ..., x_P\right], X \in \mathbb{R}^{N \times P}. \tag{2.4}$$

PCA computes the principal components by computing the eigen-decomposition of C as

$$CU = U\Lambda, \tag{2.5}$$

where U, Λ are the eigenvector and eigenvalue matrices, respectively, and the covariance matrix C is of rank equal to min (P,N). Note that computing the eigen-decomposition of C is computational expensive when the number of samples P is much smaller than the dimensionality N [Liu, 1998]. The computation load can be significantly decreased by computing instead the eigen-decomposition of the matrix C_0 defined as:

$$C_0 = \frac{1}{P} X^T X, C_0 \in \mathbb{R}^{P \times P}. \tag{2.6}$$

The eigenvectors associated to the first K dominant eigenvalues are selected to define the projection matrix P as

$$P = [u_1, u_2, ..., u_k] = [u_1, u_2, ..., u_{\min(P,N)}] \subseteq U. \tag{2.7}$$

Next, the data are projected onto the principal components subspace, namely the K selected eigenvectors, resulting in a new feature data set with lower dimensionality $(K << N)$ computed as

$$Z = P^T X, Z \in \mathbb{R}^{K \times P}. \tag{2.8}$$

c. Classification

Once the features are defined, the classifier may be derived. In this study we consider a nearest-neighbor classifier. Class-centroids based on the transformed training dataset Z are computed as

$m_{c_i} = \frac{1}{Pc_i} \sum_{j=1}^{Pc_i} z_j$, where Pc_i refers to the number of samples in the i^{th}class. (2.9)

Next, class decision for each testing image is obtained by computing the distance between the features set obtained from the testing set and all class-centroids obtained from the training set, and selecting as class that leading to the smallest distance. Different types of distances can be used, the most common and simple one to implement is the Euclidean distance (norm-2).

The PCA algorithm is summarized below:

- Reshape the number of P labeled images forming the training set of dimension $m \times n$ into a column format $N \times 1$, where $N = mn$
- Center the data set by removing the overall C classes mean,
- Form the training dataset covariance matrix,
- Compute the eigen-decomposition of the covariance matrix,
- Select the K dominant eigenvalues and associated eigenvectors (referred to as the "top K eigenvectors" in the following) for the projection step. The features space is spanned by the K top eigenvectors (i.e., principal components),
- Project the data onto the principal components space,
- Compute the class-centroids from the training data,
- Select a distance metric and compute distances between each projected image to each of the class-centroids,
- Assign a class label to each testing image by selecting as class that leading to the smallest distance between the projected testing image information and all class-centroids.

C. LINEAR DISCRIMINANT ANALYSIS (LDA)

1. Introduction

The Linear Discriminant Analysis (LDA) algorithm is a feature extraction technique commonly applied in face recognition tasks. The basic idea behind the LDA approach lies in selecting a projection so that projected class centers are far apart while the spread within each projected class is kept small. Thus, the overlap between classes is expected to be small [Schölkopf and Smola, 2002]. Results have shown that the LDA method extracts the most discriminating features, unlike the PCA that selects the most representative ones.

2. Algorithm Description

In order to introduce the underlying theory of the LDA algorithm let us illustrate the problem, as considered in [Duda, 2001], of projecting n two-dimensional data $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ onto a one-dimensional line in the direction of the vector $\mathbf{w}$ for the case of two classes, C_1 and C_2, with n_1 and n_2 samples, respectively. Samples of the two classes projected onto two different directions are depicted below in Figure 2.1. Note that the plot shown on the right of Figure 2.1 leads to better separation between the projected classes.

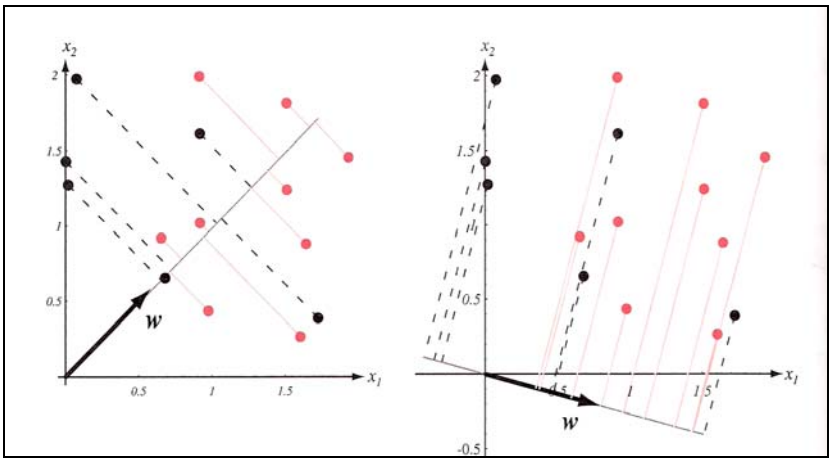

Figure 2.1 Separating two classes of a simple two-dimensional dataset. Seeking directions that are efficient for discrimination, the LDA approach (from [Duda, 2001]).

Figure 2.1 shows that varying the orientation of the projection vector $\mathbf{w}$ has a direct impact on the separation between projected samples. Thus, we need to derive a general approach that enables us to find the best projection vector $\mathbf{w}$. Note that the difference between projected sample means $\tilde{m}_i$, where

$$\tilde{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{w}^T \mathbf{x}, \; i = 1, 2, \tag{2.10}$$

can be considered a separation criterion between the two classes. In fact, a large difference implies well-separated classes, provided the projected class variances are small. However, merely increasing the difference $\left|\tilde{m}_1 - \tilde{m}_2\right| = \left|\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)\right|$ by rescaling $\mathbf{w}$ is not adequate to obtain a good separation between projected samples. We also need

to consider the class spreads. The scatter for the projected samples is a measure of the variation for each class that enables us to make such a relative comparison between the distance of the projected means and the variation of the available data.

The scatter $\tilde{s}_i$ is defined as

$$\tilde{s}_i = \sum_{\mathbf{x} \in C_i} (\mathbf{w}^T \mathbf{x} - \tilde{m}_i)^2, i = 1, 2. \tag{2.11}$$

Now a meaningful cost function that can be used to quantify the separation between two classes can be defined as follows:

$$J = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}. \tag{2.12}$$

Next, Eq. (2.12) can be explicitly expressed in terms of the projection direction vector $\mathbf{w}$ for the *N*-dimensional data and *C*-class case [Duda, 2001]. The cost function *J* is derived by introducing the scatter matrices, namely the *within-class scatter matrix* S_w, defined as:

$$S_w = \sum_{i=1}^{C} S_i, \text{where } S_i \text{ is the } \textit{per class scatter matrix} \text{ defined as} \tag{2.13}$$

$$S_i = \sum_{x \in C_i} (x - m_i)(x - m_i)^T, m_i \text{ is } \textit{the mean image} \text{ of } i^{th} \text{class} \tag{2.14}$$

and the *between-class scatter matrix* S_B defined as:

$$S_B = \sum_{i=1}^{C} n_i (m - m_i)(m - m_i)^T, \tag{2.15}$$

where m is the mean of the training images and n_i represents the number samples in the i^{th} class. Note that for the *C*-class case "the natural generalization of Fisher's linear discriminant involves *C-1* discriminant functions" [Duda, 2001]. Thus, the cost function J can be rewritten in terms of S_w, S_B and *W*, the matrix of the *C-1 N*-dimensional feature vectors $\mathbf{w}_i$, and is known as the generalized Rayleigh quotient, defined as:

$$J(W) = \frac{W^T S_B W}{W^T S_W W}. \tag{2.16}$$

The problem now becomes to find the transformation matrix *W*, so that the ratio of the between-class scatter to the within-class scatter is maximized.

It is well known that maximizing the Rayleigh quotient is obtained by computing the eigenvector associated with the maximum eigenvalue for the generalized eigen-problem [Duda, 2001], i.e.

$$S_B w_i = \lambda_i S_W w_i, i = 1, 2, ..., C-1, \quad (2.17)$$

provided that the within-class scatter matrix S_w is invertible. At this point it is interesting to note that the dimension of the new basis defined by the matrix W is a function of the total number of classes only, thus the projection is done from a *N*-dimensional to a *C-1* dimensional space.

A few observations about the nature of the within-class scatter matrix S_w are in order.

- First, in most face recognition (FR) tasks the number of training samples P is much smaller than their dimensionality N ($N >> P$), which is referred to as the so-called "small sample size" (SSS) problem, and results in most cases into a singular matrix S_w [Lu, 2003]. Thus, for a data set of P *N*-dimensional training samples representing C classes, the *N*-dimensional matrix S_w has at most $P - C$ non-zero eigenvalues. Thus, the S_w matrix is singular when *N* is greater than the matrix rank ($N > P - C$). A commonly used procedure implemented to tackle the singularity issue is to add a PCA step to remove the null space of the matrix S_w prior to the maximization of the Rayleigh quotient criterion function.
- Second, the *N*-dimensional between-class scatter matrix S_B has maximum rank equal to *C-1* because it is generated as the sum of *C* matrices of rank one, and only *C-1* of those are independent (see Appendix A, [Domboulas, 2004]).

Thus, a PCA step is first applied to reduce the dimensionality of the problem and ensure that the resulting between-class scatter matrix is non-singular, prior to applying the LDA step. The resulting scheme is known as the "Fisherface Approach."

D. BAYES CLASSFIER APPROACH

1. Introduction

Pattern recognition schemes based on Bayes decision theory assign an unknown pattern to the most probable class. That type of approach stems from the statistical variation of the collected patterns and the statistical nature of the generated features [Theodoridis, 2003]. The basic idea behind Bayesian classifiers is that the per-class

collected multidimensional samples are distributed according to a known (usually selected as the multivariate Gaussian) probability density function (PDF). Note that the Bayes classifier yields the minimum error, provided the underlying (PDFs), discussed next, are known. Bayesian error is the optimal measure for feature effectiveness in classification tasks, since it measures the class separability [Liu, 1998].

A formulation of such a classifier combines knowledge of the relative frequency of the instances of a class, namely the *a-priori* probability, with a measure of the probability that an observed feature occurs in a class, the *likelihood* probability (obtained from the training data). Those serve as inputs to the *Bayes rule*. Thus, for the testing data we compute the *a-posteriori* probability reflecting the more likely class to which unlabelled patterns belong. In the following we will define and quantify the term "more likely" based on the *Bayes rule*.

Note that the PCA will be applied for dimensionality reduction first, prior to any techniques for estimating probability density functions to ensure that the resulting per-class covariance matrices are non-singular.

2. Algorithm Description

Let us consider a set of labeled data (training data) $\{x_1,...,x_N\}$ represented by l features, collected from C different classes. For a given class $\omega_t, i=1,2,..,C$, each of the observed l-features of the vector $\boldsymbol{x}$ (i.e., pattern) is considered a random variable whose distribution depends on the nature of the classification task. For instance, if we describe each pattern $\boldsymbol{x_i}$, using l-features (i.e., $\boldsymbol{x}_i=[x_{i1},...,x_{il}]$), then $\boldsymbol{x_i}$ can be considered a multivariate Gaussian distributed variable, provided the extracted features are Gaussian distributed and independent, (see Appendix A). For classification problems the feature extraction is a crucial issue. Ideally speaking, features are extracted from the available data such that "features are different for patterns in different classes but as similar as possible for patterns in the same class" [Duda, 2001]. Thus, in probabilistic terms, different classes will be represented using the class-conditional PDFs, $p(x\,|\,\omega_t), i=1,2,..,C$.

In classification problems we need to assign an unlabeled pattern (i.e., testing data sample) represented by the l-features vector, x, to a class out of a set of C classes $\omega_1, \omega_2, ..., \omega_C$. In fact, the goal is to classify that pattern based on the C conditional probabilities, namely the *a-posteriori probabilities*, $P(\omega_i \mid x), i = 1, 2, ..., C$. Recall that each *a-posteriori probability* represents how likely the unlabeled pattern is to belong to the respective class ω_i, given that the associated feature vectors take the value x [Thedoridis, 2003]. In other words, using Bayes' rule, we assign the class label ω_i to an unlabelled data sample x when the associated posterior probability $P(\omega_i \mid x)$ is maximum over all classes. In the following we introduce a discriminant function based on the Bayes rule that is evaluated for each of the C classes, so that any unlabeled pattern is assigned to the class corresponding to the maximum of these values.

Recall that the Bayes theorem states that for a C-class problem

$$P(\omega_i \mid x) = \frac{p(x \mid \omega_i) P(\omega_i)}{p(x)}, i = 1, ..., C, \tag{2.18}$$

where ω_i represents the i^{th} class, and $p(x \mid \omega_i), i = 1, 2, ..., C,$ denotes the class-conditional PDF for the variable *x*. The PDF $p(x \mid \omega_i)$ is called the likelihood function of ω_i with respect to x and is estimated from the training data set. A-priori probabilities $P(\omega_i)$ are assumed to be known in most cases or estimated as the ratio of the per-class available training samples over the number of the total collected samples. Last, the PDF of x defined as *p(x)* can be computed as:

$$p(x) = \sum_{i=1}^{C} p(x \mid \omega_i) P(\omega_i). \tag{2.19}$$

Next, the Bayes classification rule for an unlabeled pattern represented by the feature vector x can be stated as

$$\text{Assign } x \text{ to the } i^{th} \text{ class if } P(\omega_i \mid x) > P(\omega_j \mid x), \forall\, j \neq i\ . \tag{2.20}$$

In [Thedoridis, 2003] it is shown that the Bayesian classifier minimizes the probability of classification error in the two-class problem, and that result also holds true for the generalized multi-class case.

Note that minimizing the probability of the classification error via the Bayesian probabilistic scheme can be considered as partitioning the feature space into C regions for the C-class task. In the multidimensional feature space those decision thresholds are represented by surfaces described by the equation

$$P(\omega_t \mid x) - P(\omega_j \mid x) = 0. \tag{2.21}$$

In many cases one may not deal with the posterior probability expressions directly, but with equivalent functions, referred to as "discriminant functions" $g_i(x), i = 1, 2, .., C.$ For instance, possible discriminant functions may be defined as

$$\begin{aligned} g_i(x) &= \frac{p(x \mid \omega_t)P(\omega_i)}{\sum_{i=1}^{C} p(x \mid \omega_t)P(\omega_i)} \\ g_i(x) &= p(x \mid \omega_t)P(\omega_i) \\ g_i(x) &= \log p(x \mid \omega_t) + \log P(\omega_i) \end{aligned} \tag{2.22}$$

and lead to identical classification results [Duda, 2001]. Note that any multiplication of the discriminant by a positive scalar or any addition of a constant will not affect the overall decision, and any monotonically increasing function f such that $g_i(x) \equiv f(g_i(x))$ can be used [Duda, 2001]. That choice enables us to obtain a canonical form for classifiers and leads to analytical and computational simplifications. Thus, the Bayesian classification rule can be restated in terms of the discriminant function as

$$\text{assign x to the } i^{th} \text{ class if } g_i(x) > g_j(x), \forall\, j \neq i. \tag{2.23}$$

The Bayesian classifier structure is based on the estimation of the conditional density, $p(x \mid \omega_t), i = 1, 2, .., C,$ from the training data. However, in many cases this estimation process is not an easy task. Among the various densities, the Gaussian density function has received the most attention, as it is computational tractable and models adequately a large number of cases [Thedoridis, 2003]. The multivariate normal class-conditional PDF of the *l*-dimensional patterns $\boldsymbol{x}$ is defined as

$$p(x \mid \omega_t) = \frac{1}{(2\pi)^{l/2}\left|\Sigma_t\right|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_t)^T \Sigma_t^{-1}(x - \mu_t)\right), i = 1, 2, ..., C, \tag{2.24}$$

where $\mu_i = E[x]$ is the mean vector of the i^{th} class and Σ_i the l-dimensional covariance matrix, defined as

$$\Sigma_i = \sum_{j=1}^{n_i} (x_j - \mu_i)(x_j - \mu_i)^T, \tag{2.25}$$

where n_i represents the number of training samples in the i^{th} class.

At this point we need to consider when the assumption for Gaussian distribution of the *l*-dimensional (i.e., features) patterns holds in practical cases such that equation (2.24) can be used. According to Gnanadesikan, "although marginal normality does not imply joint normality the presence of many types of non-normality is often reflected in the marginal distribution as well". Therefore in practice a preprocessing step for investigating marginal normality is applied, and multivariate assumption rejected when any marginal density is found to be not normal. Thus, a *goodness-of-fit* test may be performed based on the available training samples for each class to test for marginal normality. The normality test used in this study is the D'Agostino-Pearson K2 test proposed by D'Agostino and Stephens, [D'Agostino, 1986]. This test evaluates a dataset deviation from normality based on estimated skewness and kurtosis values. Further details regarding this test may be found in Appendix A.

The following discriminant function $g_i(x)$ can be selected when dealing with a normal PDF since it includes the monotonic logarithmic function $\ln(.)$ [Thedoridis, 2003]:

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i) + \ln P(\omega_i) - (l/2)\ln 2\pi - (l/2)\ln|\Sigma_i|. \tag{2.26}$$

Recall that the PCA step first gets applied to reduce the dimensionality of the problem and ensure that the resulting per-class covariance matrix is non-singular. Note also that the rank of the per-class covariance matrix Σ_i is a function of the number of training samples per class n_i ($rank(\Sigma_i) = n_i - 1$; see Appendix A [Domboulas, 2004]), so that the number of user-defined principal components can only be at most one less than

the rank of the covariance matrix Σ_t. Once discriminant metrics are computed for all classes, testing images are classified as belonging to the class with the maximum of these values.

Therefore, the Bayesian classifier process is implemented as follows:

- Reshape the total number of P $m \times n$-dimensional images into a $N \times 1$ column format, where $N = mn$,
- Center the data set by removing the overall C classes mean,
- Form the training dataset covariance matrix,
- Apply the PCA using the snapshoot method presented in [Pereira, 2002],
- Project the original data onto the PCA-based feature space,
- Use the projected low-dimensional training data to estimate the class-centroids and the class covariance matrices,
- Evaluate the discriminant function $g_i(x), i = 1, 2, .., C,$ for each of the testing images, and
- Assign unlabelled data sample *x* to class *i* when $g_i(x) > g_j(x), \forall\ j \neq i.$

E. ALGORITHM COMPARISONS

As mentioned earlier, the PCA approach does not consider class labels, so features extracted are better suited for representing the overall data set, unlike the LDA where the extracted features are designed to preserve separation between classes. Illustration of the projections of PCA and LDA algorithms for a toy data set are presented in Figure 2.2.

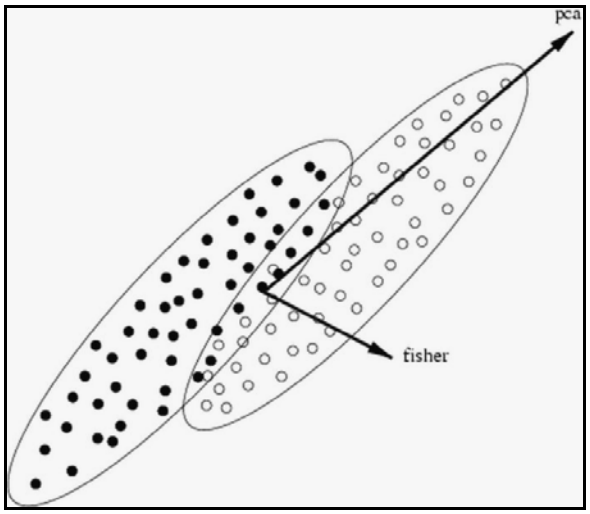

Figure 2.2 PCA and LDA projections for a toy dataset (from [Schölkopf and Smola, 2002]).

Next, some of the algorithms' main shortcomings and limitations are reviewed.

- The major issue with the PCA approach in classification tasks is the removal of the null space where most of the details useful for discrimination are encoded. As a result, some of the recent feature extraction techniques, such as [Vaswani, 2004] or [Wang, 2004], consider the null space to preserve that discriminative information.
- LDA-based classification algorithms suffer not only from the removal of the null space when the PCA preprocessing step is applied, but also from the small-sample-size problem (SSS), which occurs when the training data set is small compared to the high-dimensional feature space, resulting in instability and singularity of the within-class scatter matrix. A considerable amount of research has been devoted to tackling the SSS problem, with or without the intermediate PCA step [Lu, 2003], [Chen et al, 2000], [Yu, 2001], [Wang, 2004], [Thomaz, 2005].
- The Bayes classifier requires accurate estimations of the class density functions to provide accurate results. In most of the cases only a few training samples may be available for estimating the conditional density $p(x \mid \omega_i), i = 1, 2, .., C$. An alternative approach to the density estimation is to make an assumption of a particular density form – usually normal distribution is preferred – and estimate the associated parameters (i.e., mean and covariance matrices). Thus, the general density estimation problem is converted into a parametric one [Liu, 1998].

THIS PAGE INTENTIONALLY LEFT BLANK

III. NONLINEAR CLASSIFICATION ALGORITHMS

This chapter presents an introduction to recently proposed kernel-based schemes that provide the foundation for powerful nonlinear classification schemes. First, we briefly review some of the recent research applied to complex nonlinear classification tasks. Second, we introduce the underlying theory for kernel-based approaches along with the so called "Kernel trick." Next, we discuss the Support Vectors approach (SVs) and the Generalized Discriminant Analysis (GDA). Finally, we present a recently proposed approach to nonlinear classification tasks called the kernel-based Feature Vector Selection scheme.

A. KERNEL-BASED SCHEMES

1. Theoretical Background

Research results have shown that two widely used linear methods, namely, PCA and LDA, that can be used for low-dimension feature representation may perform poorly in pattern classification tasks when applied to complex nonlinear data sets, as a result of the linear projection constraints present in their definitions. Recent results show that kernel-based schemes lead to better classification performances, as they map the input patterns in a potentially much higher dimensional feature space where linear decision boundaries may be determined.

Kernel theory allows us to reframe linear classifiers and extend their applications to a wide range of nonlinear pattern-recognition tasks such as natural language processing, biological sequence analysis, face recognition, and text classification. Kernel theory was first discussed in the literature by Mercer in 1909 and first used for pattern recognition by Aizerman in 1964 [Baudat, 2003]. More recently, kernel theory has been used as the underlying foundation for some of the cutting-edge performance schemes in classification such as Support Vector Machines (SVM), Kernel Principal Components Analysis (KPCA), and Generalized Discriminant Analysis (GDA), also known as Kernel Fisher Discriminant Analysis (KFA).

Kernel-based schemes have been applied in a variety of research areas, including:

- Speech Technology [Kocsor, 2004]: This study focuses on the applicability of kernel-based nonlinear feature extraction and classification algorithms to the phoneme recognition in a phonological awareness drilling software package.
- Face Recognition: Numerous variations of kernel-based schemes combined with linear classification algorithms have been reported in both visible and infrared face recognition tasks, obtaining high classification rates [Baudat, 2000], [Baudat, 2003], [Liu, 2002], [Lu, 2003], [Liu, 2004].
- Text classification [Joachims, 1998], [Joachims, 1999]: SVM-based techniques have been used to organize on-line information, such as document databases, to learn users' article preferences, etc. Results have been good, even when the learning process is based on small training samples.
- 3-D Object Recognition [Pontil, 1998]: This research concentrated on the problem of recognizing 3-D objects from a single view based on SVM. Results show that high recognition rates are obtained when a small number of training samples are used, as compared to the dimensionality of the object space.
- On-line Handwriting Recognition [Bahlmann, 2002]: On-line data structure is highly complex due to the variable-size sequence of feature vectors. In this study, SVMs' discriminative power previously used with success for Optical Character Recognition (OCR) is combined with a new type of kernel function in on-line handwriting recognition tasks [Muller, 2001].

There is currently no doubt that kernel-based approaches are well suited tools for feature extraction, especially in nonlinear and high-dimensional data sets. However, both kernel function selection and parameter tuning issues remain challenging and are still open topics of research. In most cases, trial-and-error or hold-out cross-validation approaches are used for selecting parameters, which is computationally intensive and may cause over-fitting problems. Moreover, kernel matrices have dimensions equal to the number of the available training samples, potentially leading to extremely large dimensions for large data sets. Recently, a considerable amount of effort has been made to address kernel parameters optimization issues [Huang, 2004], [Xiong, 2005], [Centeno, 2006] and kernel matrix dimension reduction problems by removing "outlier" data points [Kim, 2003], [Baudat, 2003].

2. Introduction to Kernel Theory: The Kernel Trick

The kernel trick has been around for a long time. However, it was not until the mid 1990s that researchers discovered that any dot-product-based algorithm can be "kernelized." Kernel methods for pattern recognition can be considered a generalization of more limited but well-established classical linear classifiers via a mapping function ϕ. The basic idea behind kernel-based schemes is to first map the input vectors $x \in X$ into a high-dimensional feature space ***F***, in which data are nonlinearly related to the input space [Baudat, 2000]. Standard linear schemes may then be applied in the transformed space.

Increasing the number of features representing patterns while keeping the number of training samples fixed is an important issue in pattern recognition applications. In classification applications for example, the number of data samples needed to accurately estimate decision boundaries grows exponentially with the dimension of the feature space. This characteristic is better known as the "curse of dimensionality" and results in computationally expensive and memory-consuming methods being required to handle problems associated with high-dimensional features. However, efficient decision boundaries may be obtained when classes are well separated in a higher-dimensional space or any a-priori information about the data structure is known (i.e., a specific type of distribution, independence between features, etc.), [NAVSEA Newport, 2006].

Next, we introduce the crucial concept used in kernel based schemes, referred to as the "kernel trick," following the presentation given in [Schölkopf and Smola, 2002].

Let us define a dataset X of patterns $\mathbf{x}_i$ (also called observations, training samples, or inputs) with known labels $y_i \in Y$ (also called targets, or outputs) that form a training set of m examples $(x_1, y_1), ..., (x_m, y_m) \in X \times Y$. Note that, following the terminology of [Schölkopf and Smola, 2002], we will refer to the term "pattern" for an individual observation which is more common in the machine learning area.

In most machine learning tasks the goal is to find a suitable label $y \in Y$ for any new observation $x \in X$ so that (x, y) is similar to the training examples $(x_1, y_1), ..., (x_m, y_m)$. In fact, defining the similarity of outputs in a binary classification task is easy, as labels will either be the same or different. Unlike outputs, selecting

similarity measures for the inputs remains a challenging issue [Schölkopf and Smola, 2002]. Therefore, a function that returns a real number representing in a quantitative manner the similarity between any pair of input patterns is needed. A function k, called kernel, of the form

$$\begin{aligned} k &: X \times X \to \mathbb{R} \\ (x, x') &\mapsto k(x, x') \end{aligned} \tag{3.1}$$

may be used as a measure of similarity between the patterns x, x'.

A widely used similarity measure between two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$ is the dot product defined as

$$\langle \mathbf{x}, \mathbf{x}' \rangle := \sum_{i=1}^{N} [x]_i [x']_j, \text{where } [x]_i \text{ is the i}^{\text{th}} \text{ coordinate of } \mathbf{x}. \tag{3.2}$$

Note that the dot product is considered an attractive similarity measure from the mathematical perspective, since it characterizes the level of similarity between two vectors, which can be expressed in terms of angles, lengths, and distances between the vectors.

Although the dot product is a particularly simple similarity measure, it is not sufficient for the case of non-vectorial input data. Applications with non-vectorial input data (also referred as structured data) are usually met in bioinformatics and may include protein classification (x_i is a string of amino acids, or a molecule structure) or in information extraction (x_i is a sentence of words) [Joachims, 2006]. A powerful property of kernel theory is used when dealing with structured data configurations to convert structures into real vectors and then implement learning algorithms. The representation of patterns as vectors is done via the following mapping function ϕ, defined as:

$$\begin{aligned} \phi &: X \to \boldsymbol{F} \\ x &\mapsto \mathbf{x} := \phi(x). \end{aligned} \tag{3.3}$$

Overall, kernel theory allows the transformation of data in a transformed space $\boldsymbol{F}$, referred to as the "dot product space" or "feature space," via a nonlinear mapping ϕ, where the transformed data can be treated efficiently using linear classification methods.

The main benefits of mapping the original data via ϕ are summarized below [Schölkopf and Smola, 2002]:

1. Enable to define a similarity measure in the feature space F as

$$k(x, x') := \langle \mathbf{x}, \mathbf{x}' \rangle = \langle \phi(x), \phi(x') \rangle. \tag{3.4}$$

2. Patterns are treated geometrically. Thus, linear algebra and analytic geometry methods may apply for learning schemes.

3. Adaptability in selecting the mapping function ϕ is best suited for overlapping classes, thus potentially resulting in a nonlinear mapping used to transform an initially complex representation space into a linear separable space, where linear classification schemes may then be applied.

Next, we consider a simple example where a kernel-based approach is used to compute dot products in spaces of monomial features to illustrate the usefulness of kernels and introduce the "kernel trick."

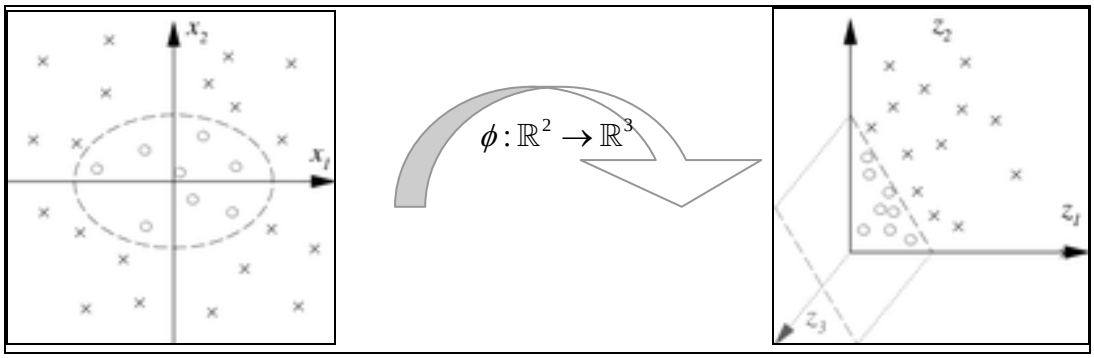

Figure 3.1 Binary classification example. Note: The dot product in the three-dimensional space can be computed without computing the mapping function ϕ, (after [Schölkopf and Smola, 2002]).

Let us consider a pair of inputs $x, y \in \mathbb{R}^2$ that are nonlinearly transformed using the mapping function ϕ as follows:

$$\begin{aligned} &\phi : \mathbb{R}^2 \to \mathbb{R}^3 \\ &(x_1, x_2) \to (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2), \end{aligned} \tag{3.5}$$

where the transformed input patterns *x* and *y* may be expressed in the transformed space as *a* and *b* defined, as shown below:

$$
\begin{aligned}
&x = (x_1, x_2), y = (y_1, y_2), \\
&a = \phi(x), b = \phi(y), \\
&a = (a_1, a_2, a_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2), \\
&b = (b_1, b_2, b_3) = (y_1^2, \sqrt{2}y_1y_2, y_2^2).
\end{aligned} \tag{3.6}
$$

Thus, the corresponding kernel expression $k(x, x') = \langle \phi(x), \phi(x') \rangle$ is given by

$$
\begin{aligned}
\langle \phi(x), \phi(y) \rangle = \langle a, b \rangle &= a_1b_1 + a_2b_2 + a_3b_3 \\
&= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\
&= \langle x, y \rangle^2 ,
\end{aligned} \tag{3.7}
$$

which shows that the kernel expression is equivalent to

$$
k(x, y) = \langle x, y \rangle^2 \tag{3.8}
$$

in the input space. Numerous types of kernel functions may be used, depending on the data characteristics. Some typical kernel functions are:

Gaussian type: $k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2} \right)$, where $\sigma > 0$, (3.9)

Sigmoid type: $k(x, y) = \tanh(a\langle x, y \rangle + b)$, where $a > 0, b < 0$, (3.10)

Polynomial type: $k(x, y) = (a\langle x, y \rangle + b)^d$, $a, b, d > 0$. (3.11)

Note that a kernel k must satisfy specific properties to correspond to some nonlinear transformation ϕ and ensure that $k(x, x') = \langle \phi(x), \phi(x') \rangle \forall x, x'$. The constraints are summarized below [Schölkopf and Smola, 2002].

1. the mapping $k(x, x')$ is continuous,
2. the mapping k is symmetric, i.e., $k(x, x') = k(x', x) \forall x, x'$,
3. the mapping $k(x, x')$, is positive definite, i.e., $\int k(x, x')g(x)g(x')dxdx' \geq 0, \forall g$.

Kernel concepts may be extended to spaces other than dot product spaces, where a vectorial representation of objects is not readily available, including data types such as text strings, etc. [Schölkopf and Smola, 2002].

3. Support Vectors

Support Vectors classification (SVC) schemes are widely applied in pattern recognition tasks [Joachims, 1998], [Muller, 2001], [Pontil, 1998]. The basic idea behind SVC schemes is the generation of the separating hyperplane that linearly separates data in a classification task, as shown in Figure 3.2.

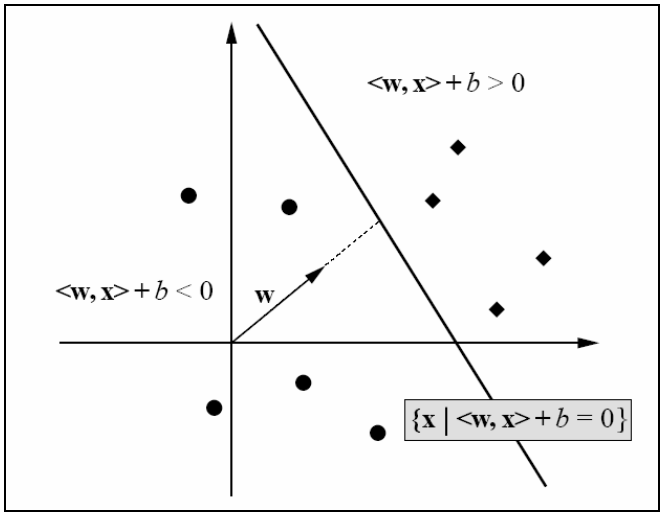

Figure 3.2 A separable classification problem along with a separating hyperplane written in terms of an orthogonal weight vector **w** and a threshold *b*. (from [Schölkopf and Smola, 2002]).

It has been shown that only one separating hyperplane out of all possible separating hyperplanes defined for a given classification problem is of maximum margin to any training point. The term "margin" refers to the distance to the separating hyperplane from the closest point to it, and further details may be found in [Scholkopf and Smola, 2002, Chapter 7]. Note that a large margin is usually needed for a learning algorithm to generalize well [Schölkopf and Smola, 2002].

Computing the optimal hyperplane comes down to solving a constraint optimization problem (refer to [Schölkopf and Smola, 2002, Chapters 6 and 7] for a detailed analysis). For a set of training vectors $\mathbf{x} \in \mathbb{R}^N$, the solution to the optimal margin classifier problem, referred to as the vector **w**, can be decomposed in terms of a subset of the training patterns lying on the margin of the optimal hyperplane, which are called the Support Vectors (SVs). Hence, the optimal hyperplane defined by the vector $\mathbf{w} \in \mathbb{R}^N$ and

the constant $b \in \mathbb{R}$ satisfying $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ is defined. As a result, training patterns not used in the expansion of the vector $\mathbf{w}$ can be ignored, resulting in only the patterns closest in distance to the optimal hyperplane encountered needed to determine class boundaries. These specific patterns are located on what are referred to as "support hyperplanes," as depicted in Figure 3.3.

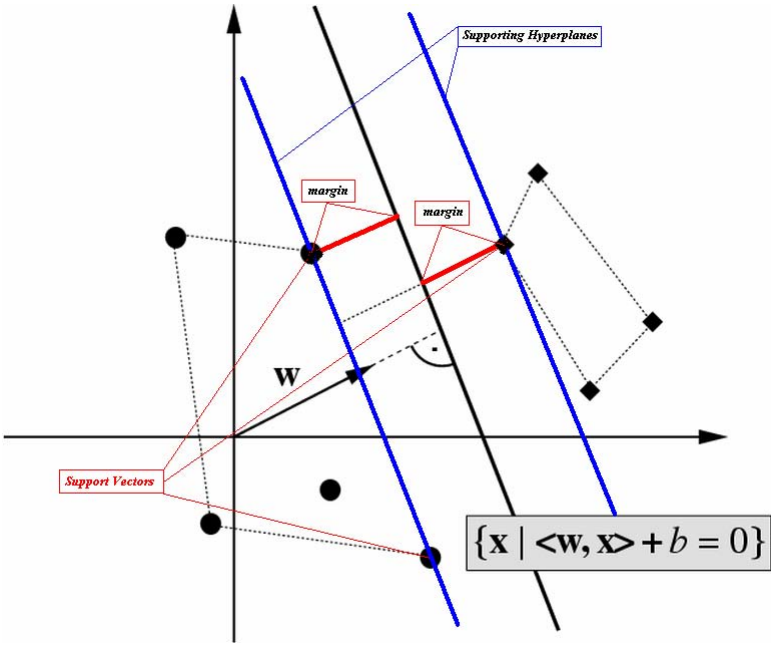

Figure 3.3 The optimal hyperplane for a two-class problem, bisects the shortest connection between the convex hulls of the two classes. (after [Schölkopf and Smola, 2002]).

The SVC approach may be generalized to classify nonlinear separable data by applying kernel concepts. In such a case, a kernel preprocessing step needs to be incorporated to transform the input data into a high-dimensional nonlinear space where the original nonlinearly separable problem may be reframed as a separable one, as illustrated in Figure 3.4.

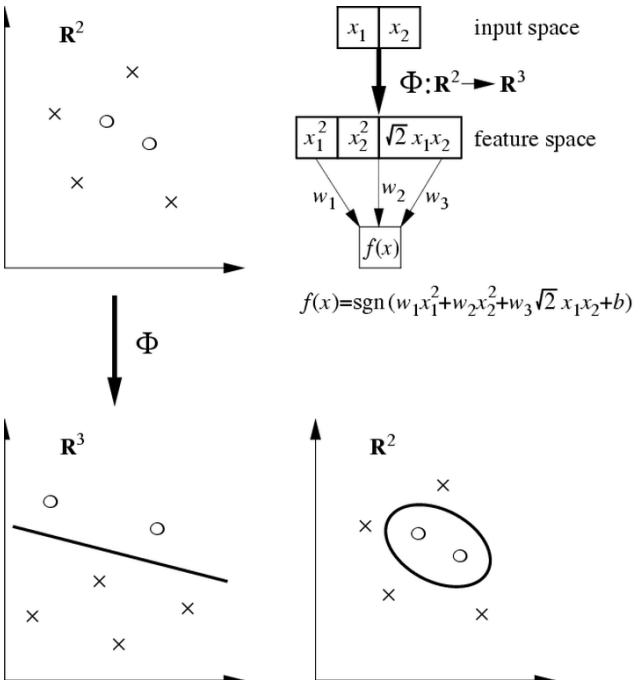

Figure 3.4 SVM "kernelizing" the optimal margin hyperplane (from [Schölkopf and Smola, 2002]).

4. Generalized Discriminant Analysis

Nonlinear kernel-based classification algorithms are implemented in two steps. First, a nonlinear kernel function is selected to map the patterns from the complex input space into a linear high-dimensional feature space. Next, classic linear methods such as PCA and LDA may be applied in the new feature space, resulting in the KPCA [Schölkopf and Smola, 2002] and Generalized Discriminant Analysis (GDA) approaches [Baudat, 2000], respectively. The main advantage behind the kernel based schemes is that there is no need to explicitly derive the mapping function, as the transformation to the new feature space is done implicitly via the "kernel trick."

Note that reframing the original problem into a high-dimensional space results in an approach with a high computational load, resulting from storing and processing the kernel matrix, defined from the kernel function as

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j),\ i, j = 1, ..., m, \tag{3.12}$$

where $\mathbf{x}_i \in \mathbb{R}^N$ are the m available training data (examples) and k is a valid kernel function. As a result, Anouar and Baudat recently proposed an alternative to the preprocessing step used in the GDA scheme, which selects only a relevant portion of the

available data, thereby significantly reducing the scheme computational load. Next, we introduce the GDA approach and present this extension, called Feature Vector Selection LDA [Baudat, 2003].

Assume that they are m training data $\mathbf{x}_i \in \mathbb{R}^N, i = 1,...,m$ available. Define a nonlinear mapping $\phi : \mathbf{x}_i \in \mathbb{R}^N \rightarrow \phi(\mathbf{x}_i) \in \boldsymbol{F}^M, M >> N$ that transforms input patterns into a high-dimensional space $\boldsymbol{F}$. The GDA approach is defined so that the classic LDA method is applied to the nonlinearly transformed data in the feature space $\boldsymbol{F}$. Thus, the GDA implicitly maximizes the ratio of the between-class and within-class scatter matrices, where S_B and S_W scatter matrices are defined in the transformed space as:

$$\begin{aligned} S_B &= \frac{1}{m}\sum_{i=1}^{C} m_i(\bar{\phi}_i - \bar{\phi})(\bar{\phi}_i - \bar{\phi})^T \\ S_W &= \frac{1}{m}\sum_{i=1}^{C}\sum_{j=1}^{m_i}(\phi_{ij} - \bar{\phi}_i)(\phi_{ij} - \bar{\phi}_i)^T, \end{aligned} \tag{3.13}$$

where m represents the total number of training samples, m_i is the samples in class i, C is the number of classes, $\phi_{ij} = \phi(\mathbf{x}_{ij}), \bar{\phi}_i = \frac{1}{m_i}\sum_{j=1}^{m_i}\phi(\mathbf{x}_{ij})$ is the mean vector of class i, and $\bar{\phi} = \frac{1}{m}\sum_{i=1}^{C}\sum_{j=1}^{m_i}\phi(\mathbf{x}_{ij})$ is the average of the mean vectors over all classes C.

B. FEATURE VECTOR SELECTION AND PROJECTION USING KERNELS

1. Introduction

The Feature Vector Selection (FVS) approach addresses the issue of the large dimensional feature space $\boldsymbol{F}$ by selecting a subspace $\boldsymbol{F_S}$ as a basis to represent the available data. Recall that Kernel-based algorithms use kernel matrices of dimensions equal to the number of training samples resulting in potentially complex and high computational cost solutions. FVS is designed to dramatically reduce memory requirements, by dealing with kernel matrices of a size equal to the number of features selected to span the subspace $\boldsymbol{F_S}$. Moreover, the size of the subspace $\boldsymbol{F_S}$ is related to the model complexity; thus complexity control can be obtained via data selection [Baudat, 2003]. FVS is considered a data-preprocessing step, and linear algorithms may apply to the transformed data expanded in terms of the basis spanning $\boldsymbol{F_S}$.

2. Algorithm Description

This section discusses the FVS algorithm and closely follows the presentation given by [Baudat, 2003]. Let ϕ be a mapping function that transforms patterns from an input space *X* to a feature Hilbert space ***F****:*

$$\begin{aligned} \phi &: \mathrm{X} \rightarrow \boldsymbol{F} \\ x &\rightarrow \phi(x). \end{aligned} \tag{3.14}$$

Recall the kernel matrix for a set of *M* training samples is defined as

$$K = (k_{ij})_{1 \le i,\, j \le M}, \text{where } k_{ij} = \phi^T(x_i)\phi(x_j). \tag{3.15}$$

The dimensionality of the subspace $\boldsymbol{F_S}$ is related to the size of the kernel matrix *K* and is selected as the rank of the kernel matrix *K,* which may be much lower than *M*. The proposed method captures the geometrical structure of the training data set in the feature Hilbert space ***F***, extracting the most relevant data, called feature vectors (FVs), that span the reduced subspace $\boldsymbol{F_S}$. Those selected FVs are the basis onto which all the remaining data are linearly projected.

Training samples x_i are mapped in the transformed space using the mapping function ϕ, resulting in mapped data $\phi_i = \phi(x_i)_{1 \le i \le M}$, where *M* represents the total number of samples in the input space. One can select a subset *L* of these training samples, denoted as x_{si}, with associated transformed samples $\phi_{si} = \phi(x_{si})_{1 \le i \le L}$, where $L \le M$. The selected *L* samples are referred to as the "feature vectors" (FVs). Recall that the basic idea behind the FV approach is to express mapped samples in terms of this specific set of FVs in the transformed space. Thus, the estimated transformed mapping of any training vector x_i can be expressed as:

$$\hat{\phi}_i = \Phi_S \cdot a_{i,} i = 1, ..., M, \tag{3.16}$$

where $\Phi_S = (\phi_{S1}, ..., \phi_{SL})$ is the matrix of the selected set of vectors $X_S = \{x_{S_1}, ..., x_{S_L}\}$, $S = \{s_1, ..., s_L\}$ and $a_i = (a_{i1}, ..., a_{iL})^T$ is the coefficient coordinates vector.

Thus, the FVs selection problem comes down to extracting a set *S* of *L* feature vectors (and associated coefficients a_i), such that the estimated mapping $\hat{\phi}_i$ is a good approximation of the real mapping ϕ_i. This problem can be formulated as minimizing the following normalized Euclidean distance:

$$\delta_t = \frac{\left\|\phi_t - \hat{\phi}_t\right\|^2}{\left\|\phi_t\right\|^2}. \tag{3.17}$$

The ratio δ_t is a measure of the angle distance θ_i between the estimated mapping $\hat{\phi}_i$ and the real mapping ϕ_i. Note that the ratio δ_t evaluates how close in direction the vectors ϕ and $\hat{\phi}_i$ are, since δ_t is equal to $\sin^2(\theta_i)$, where θ_i represents the angle between the vectors ϕ and $\hat{\phi}_i$ [Baudat, 2003]. Substituting (3.15) and (3.16) in (3.17) leads to:

$$\delta_i = \frac{(\phi_i - \Phi_S \cdot a_i)^T (\phi_i - \Phi_S \cdot a_i)}{\left\|\phi_t\right\|^2}. \tag{3.18}$$

Next, differentiating δ_t with respect to a_i leads to

$$\begin{aligned} &\frac{\partial \delta_i}{\partial a_i} = \frac{2(\Phi^T{}_S \Phi_S) a_i - 2(\Phi^T{}_S \phi_i)}{(\phi_i^T \phi_i)} = 0, \\ &a_i = (\Phi^T{}_S \Phi_S)^{-1} \Phi^T{}_S \phi_i. \end{aligned} \tag{3.19}$$

Note that, the matrix $(\Phi^T{}_S \Phi_S)$ is non-singular, provided that the FVs are linearly independent in $\boldsymbol{F}$. Thus, replacing coefficient a_i in (3.18) leads to

$$\min_{a_i} \delta_t = 1 - \frac{(\phi_i^T \Phi_S (\Phi^T{}_S \Phi_S)^{-1} \Phi^T{}_S \phi_i)}{\left\|\phi_t\right\|^2}. \tag{3.20}$$

Baudat and Anouar showed that solving Eq. (3.20) is equivalent to solving [Baudat, 2003] $\min_{a_i} \delta_t = 1 - \frac{\vec{K}_{Si}^T K_{SS}^{-1} \vec{K}_{Si}}{k_{ii}}$, where the matrix K_{ss} is the *L*-dimensional kernel matrix computed from the *L*-selected feature vectors and defined as:

$$K_{SS} = (k_{Si,Sj})_{1 \le i,\, j \le L}, K_{ss} \in \mathbb{R}^{L \times L}, \tag{3.21}$$

K_{Si} is the vector of inner products between samples x_i and the FVs, and is defined as

$$\vec{K}_{Si} = (k_{Sj,i})_{1 \le j \le L}. \tag{3.22}$$

Thus, (3.20) may be rewritten as

$$\min_{a_i} \delta_i = 1 - \frac{\vec{K}_{Si}^T K_{SS}^{-1} \vec{K}_{Si}}{k_{ii}}. \tag{3.23}$$

Therefore, the goal is to find the set *S* of input samples that overall maximizes Eq. (3.23) for all *M* samples x_i, i.e., to find the set S which computes J_1,defined as:

$$\begin{aligned} J_1 &= \min_S \left(\frac{1}{M} \sum_{x_i \in X} \left(1 - \frac{\vec{K}_{Si}^T K_{SS}^{-1} \vec{K}_{Si}}{k_{ii}} \right) \right) \\ &= \min \left(\frac{1}{M} \sum_{x_i \in X} 1 - J_{Si} \right). \end{aligned} \tag{3.24}$$

Baudat and Anouar showed that J_{Si}, referred to as the local fitness measure, may be rewritten as [Baudat, 2003]:

$$J_{Si} = \left(\frac{\vec{K}_{Si}^T K_{SS}^{-1} \vec{K}_{Si}}{k_{ii}} \right) = \frac{\left\| \hat{\phi}_i \right\|^2}{\left\| \phi_i \right\|^2}.$$

Note that minimizing J_l is equivalent to maximizing J_S, referred to as the global fitness, where J_S is defined as

$$J_S = \frac{1}{M} \sum_{x_i \in X} J_{Si}. \tag{3.25}$$

The FVs selection method is implemented iteratively and extracts FVs close to linearly independent, thereby avoiding a potentially ill-conditioned solution [Baudat, 2003, Appendix C]. Moreover, the iterative process is computationally attractive, as it stops when the kernel matrix K_{SS} becomes singular, indicating that the selected data X_S form a well-approximated basis for the remaining data in ***F***. User-specified stopping parameters are the number of FVs (where the maximum number is equal to the total number of training samples *M*) and the maximum bound for the global fitness J_S (with the J_S maximum value equal to one). Finally, a cross-validation approach may be applied to estimate the polynomial kernel order or the variance of the exponential kernel variance parameter best suited for the data under investigation. The last step remaining in

the procedure is to map the data into the reduced subspace $\boldsymbol{F_s}$. Thus, any given input sample x_i is transformed by using the projection matrix of the selected vectors $\Phi_S = (\phi_{S1}, ..., \phi_{SL})$, leading to

$$z_i = \Phi^T{}_S \phi_i. \;\; i = 1, ...M. \tag{3.26}$$

At that point, linear classification algorithms may be applied in the transformed data set, resulting in an overall nonlinear scheme. Figure (3.5) shows the mechanism of the discussed method combined with a Linear Regression (LR) step used for a function approximation.

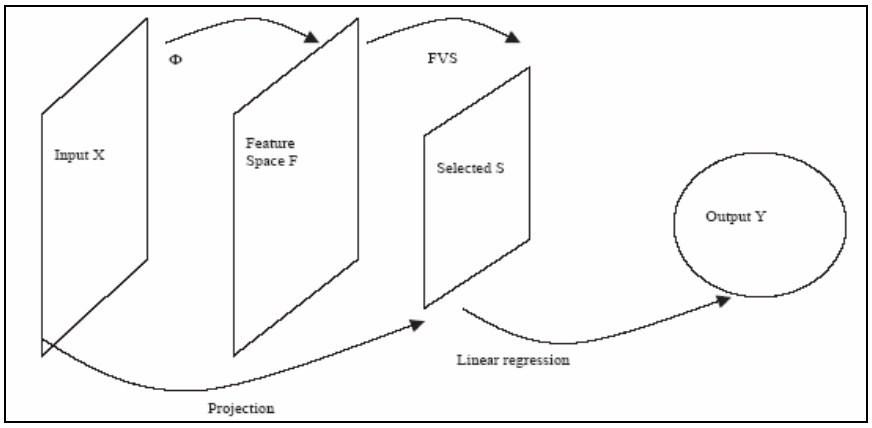

Figure 3.5 Architecture of FVS-LR (from [Baudat, 2001]).

Baudat and Anouar [Baudat, 2003] applied the FVS on a simple two-class, two-dimensional data classification problem. They generated one hundred training samples using normal distributions with the following mean and covariance matrices:

$$C_1 \sim N\left([-1,0], \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right),\; C_2 \sim N\left([1,0], \begin{pmatrix} 1 & 0 \\ 0 & 0.1 \end{pmatrix}\right), \tag{3.27}$$

and showed that results obtained with the FVS approach using 21 feature vectors and a Gaussian Kernel of unit variance were close to those obtained with the Bayesian classifier, as illustrated in Figure 3.6.

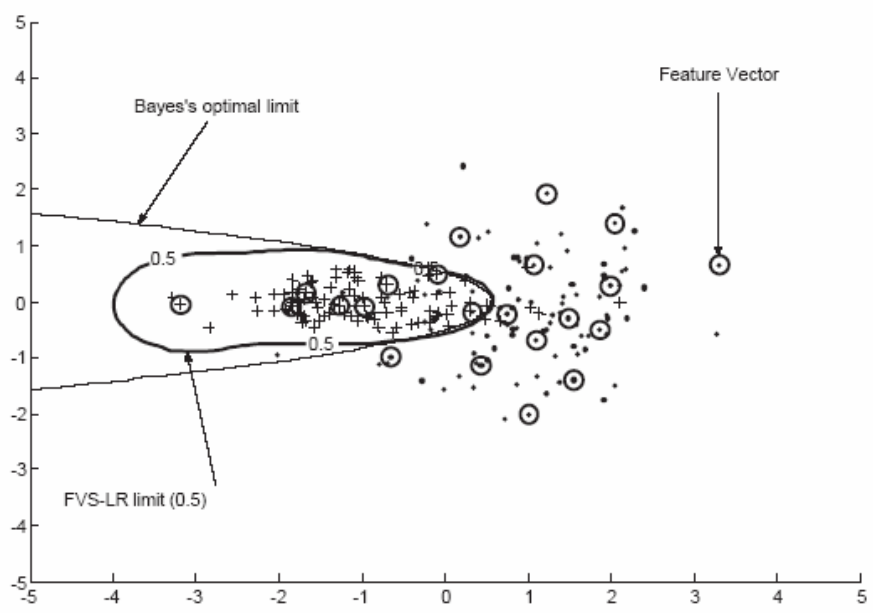

Figure 3.6 Optimal boundary and FVS-LR boundary; features are represented by circles (from [Baudat, 2003]).

C. ALGORITHMS COMPARISON

The FVS approach offers several advantages over the GDA scheme. First, memory requirements for matrix storage dramatically reduce from the size of the available training samples, as is the case for the GDA approach, to the size of the selected feature vectors only. Second, the GDA approach computes the eigen-decomposition of the potentially ill-conditioned kernel mapping matrix *K*, which is computationally expensive, while the iterative process in the FVS decomposition is designed to stop when K_{SS} is no longer invertible, thus avoiding singularity issues. Third, FVS is a data-preprocessing step that can be followed by many classic linear methods (PCA, LDA, LR, etc.). Moreover, the FVS step combined with a linear classification algorithm can be considered a good approximation of the full kernel-based version when the following orthogonal transformation is used to project in the subspace $\boldsymbol{F_S}$,

$$z_i = (\Phi^T{}_S \Phi_S)^{-1/2} \Phi^T{}_S \phi_t \qquad (3.28)$$

[Baudat, 2003]. Finally, note that applying the above transformation with all the training data as FVs leads to the kernelized versions of the algorithms [Baudat, 2003].

Next, GDA and FVS-LDA schemes are applied to the “Iris” data for comparison purposes. The “Iris” data set includes three species of the iris flower characterized by

four features, namely, the sepal length, the sepal width, the petal length, and the petal width. It is well known that two of the classes are linearly separable, while the third is not linearly separable from the other two. As a result, this dataset is commonly used in classification applications for benchmarking. A total of one hundred and fifty examples (fifty per class) are available. The Gaussian kernel function used in both schemes is $k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2} \right)$.

Two tests were implemented to investigate the effect of selecting the parameters for FVS-LDA, namely, the number of features (FVs) and the variance of the Gaussian Kernel function. First, we varied the kernel variance parameter σ^2 for a fixed number of features to investigate the impact of this parameter selection on the resulting classifier performance. Results are depicted in Figures 3.7 to 3.10. Second, we varied the number of features while keeping the kernel variance parameter σ^2 fixed, as illustrated in Figures 3.11 to 3.14. Results show that the variance parameter plays a crucial role in the resulting within-class data clustering and the between-class separation. Note that large or small values of σ result in class overlap, as shown in Figures 3.7 and 3.10. In addition, Figures 3.13 and 3.14 show that better class separation may be obtained by increasing the number of features. In fact the latter is more efficient when the variance is selected well.

Finally, these simulations also showed that FVS-based results are close to those reported in an earlier study with the GDA approach [Domboulas, 2004], where three values for the spread σ were considered. Note that the FVS-LDA approach is designed to separate the three classes as accurately as the GDA does for a dramatically reduced computational load. Figures 3.15 to 3.20 show that the FVS-LDA can be considered a close approximation to GDA, even for a significantly lower feature representation, provided that the variance for the kernel function is selected judiciously, as shown in Figure 3.20.

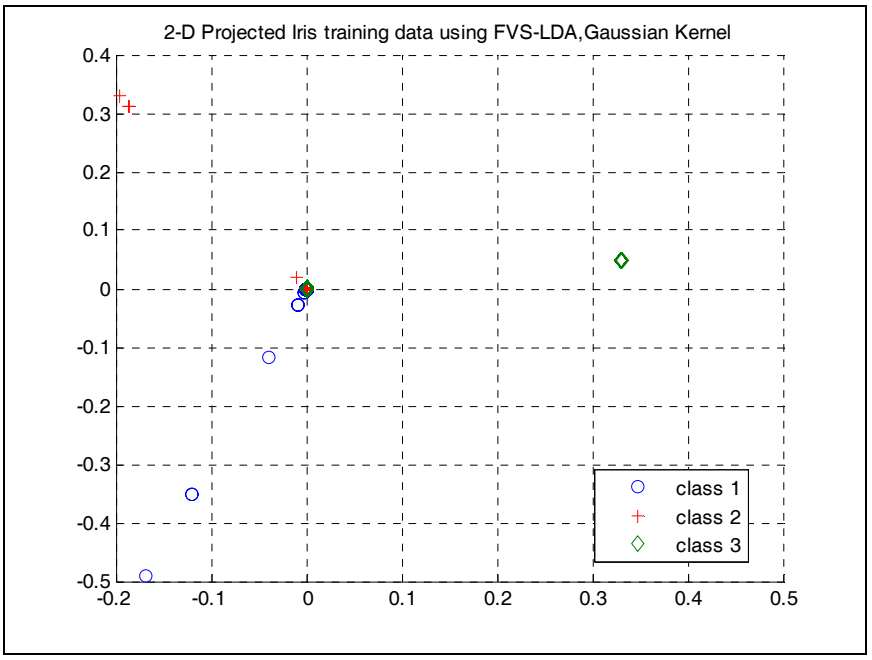

Figure 3.7 FVS-LDA projection of Iris data set, variation ($2\sigma^2$=0.007), 15 FVs selected (max 150 FVs).

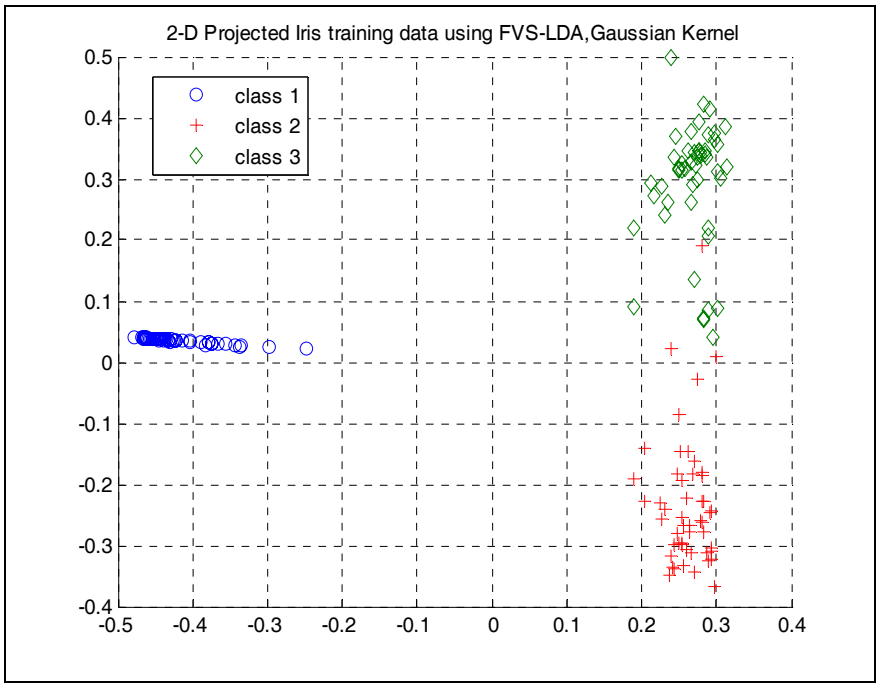

Figure 3.8 FVS-LDA projection of Iris data set, variation ($2\sigma^2$=0.7), 15 FVs selected (max 150 FVs).

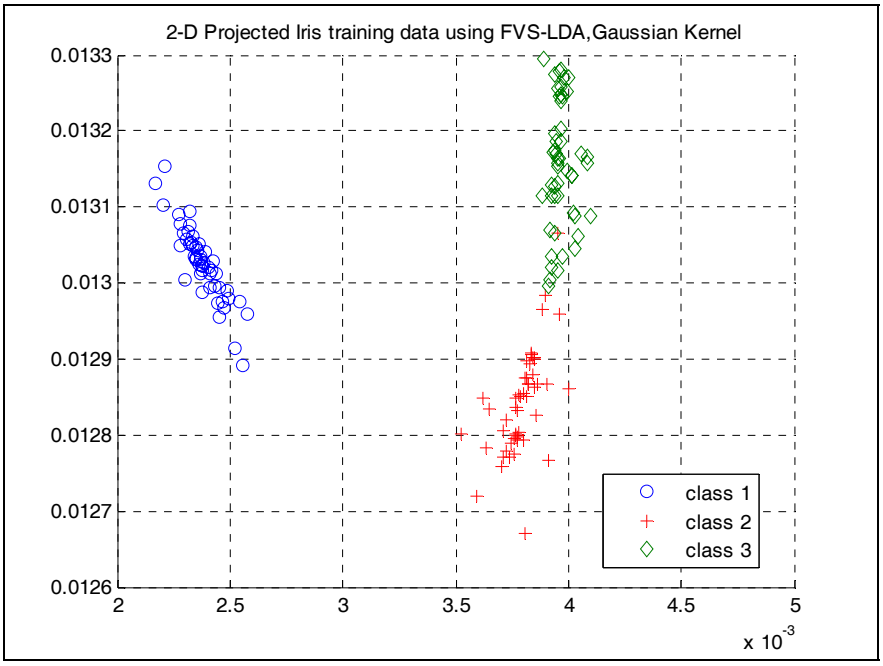

Figure 3.9 FVS-LDA projection of Iris data set variation ($2\sigma^2$=70), 15 FVs selected (max 150 FVs).

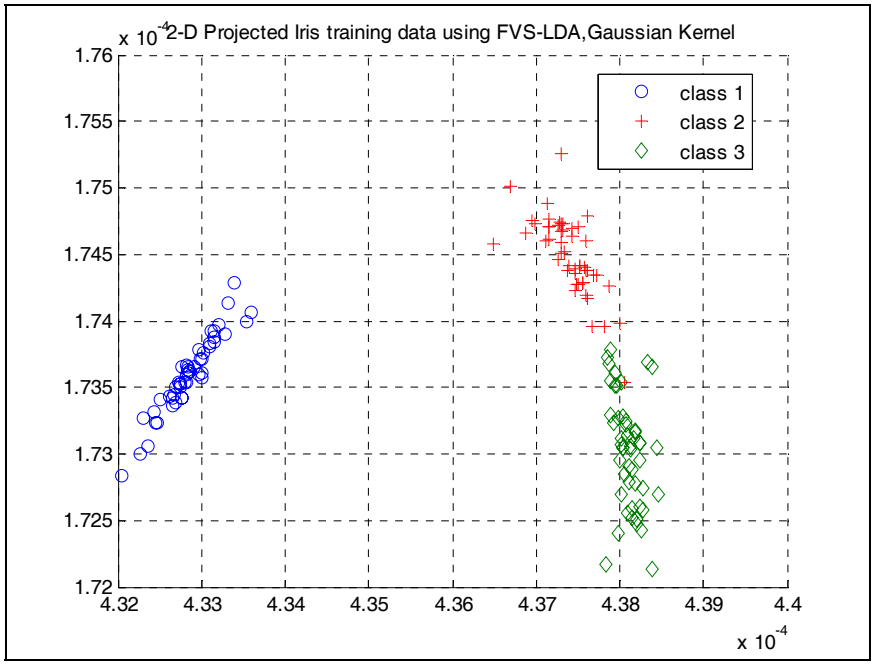

Figure 3.10 FVS-LDA projection of Iris data set, variation ($2\sigma^2$=700), 15 FVs selected (max 150 FVs).

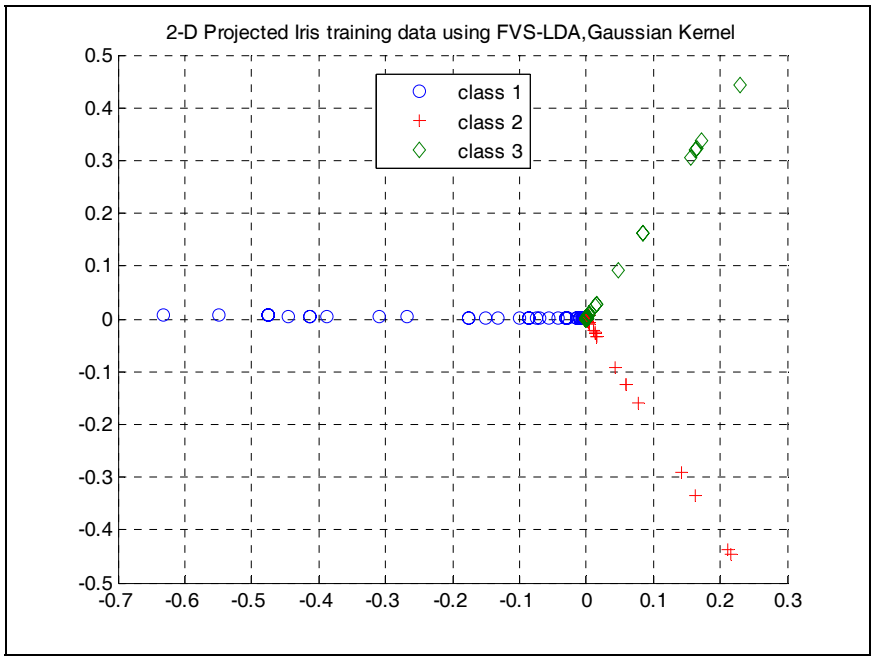

Figure 3.11 FVS-LDA projection of Iris data set, variation ($2\sigma^2$=0.07), 10 FVs selected (max 150 FVs).

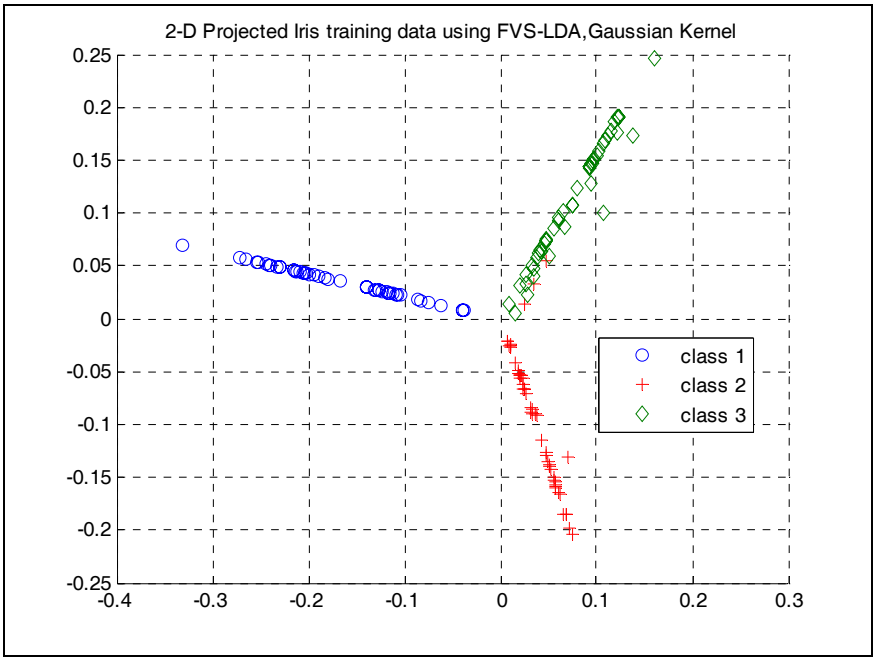

Figure 3.12 FVS-LDA projection of Iris data set variation ($2\sigma^2$=0.07), 50 FVs selected (max 150 FVs).

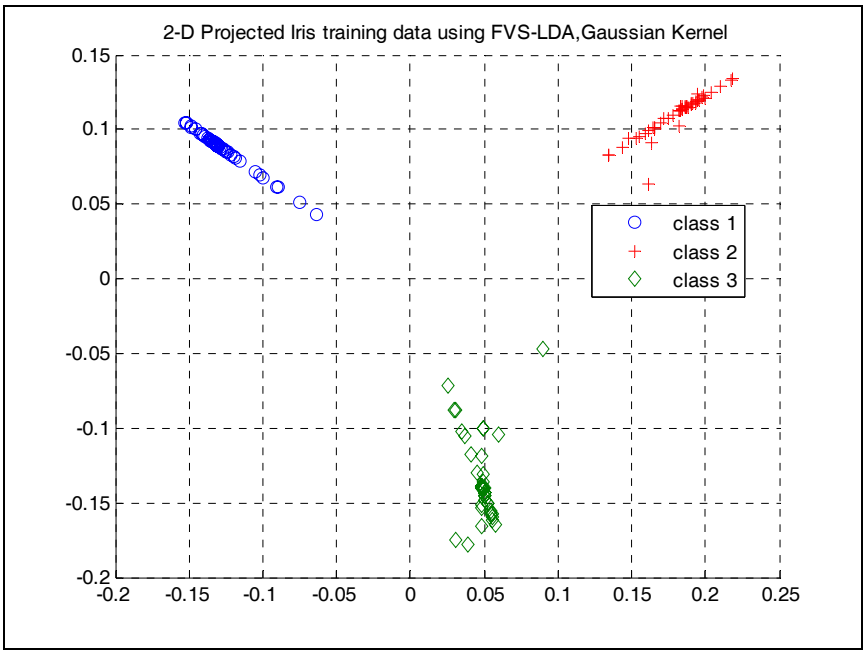

Figure 3.13 FVS-LDA projection of Iris data set variation ($2\sigma^2$=0.07), 90 FVs selected (max 150 FVs).

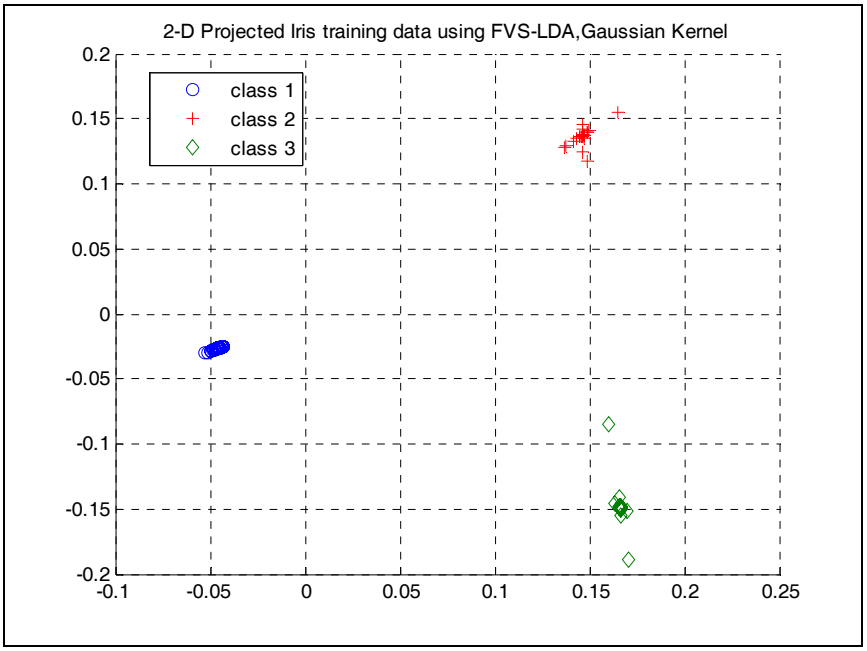

Figure 3.14 FVS-LDA projection of Iris data set variation ($2\sigma^2$=0.07), 130 FVs selected (max 150 FVs).

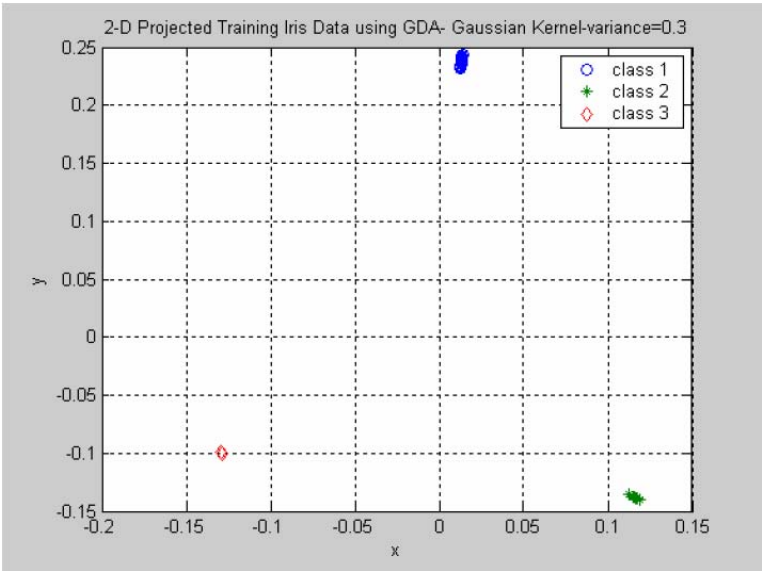

Figure 3.15 GDA projection of Iris data set, small variance ($2\sigma^2$=0.3) (from [Domboulas, 2004]).

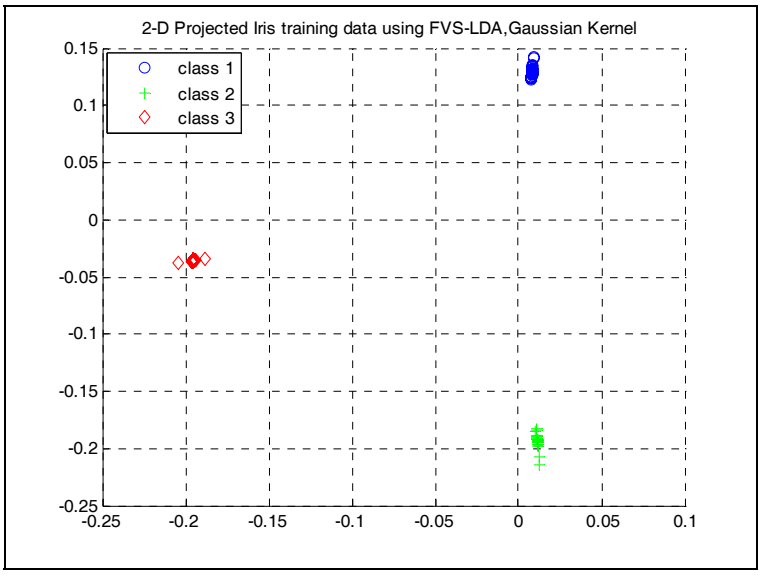

Figure 3.16 FVS-LDA projection of Iris data set, small variation ($2\sigma^2$=0.1), 130 FVs selected (max 150 FVs).

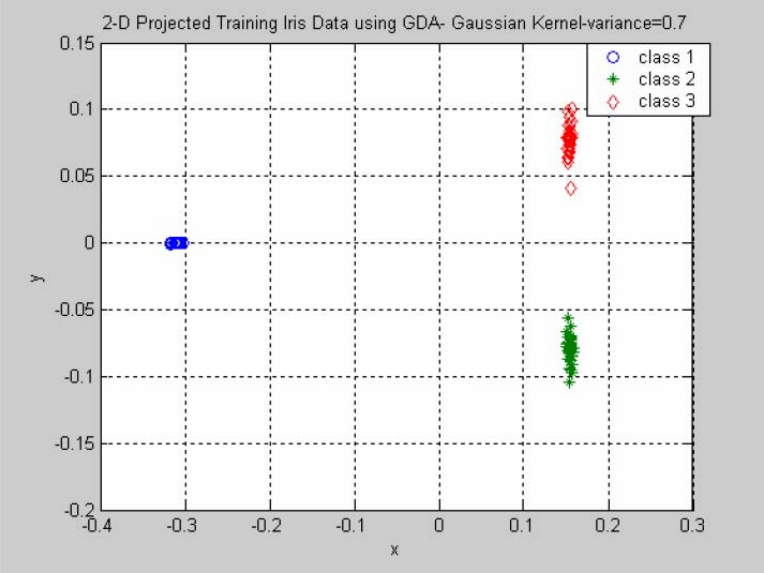

Figure 3.17 GDA projection of Iris data set, moderate variation ($2\sigma^2$=0.7) (from [Domboulas, 2004]).

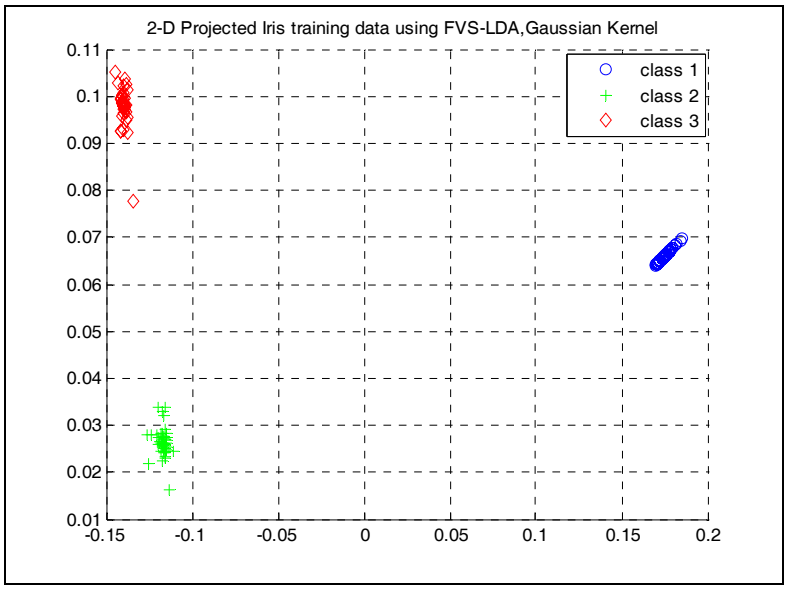

Figure 3.18 FVS-LDA projection of Iris data set, moderate variation ($2\sigma^2$=0.3), 100 FVs selected (max 150 FVs).

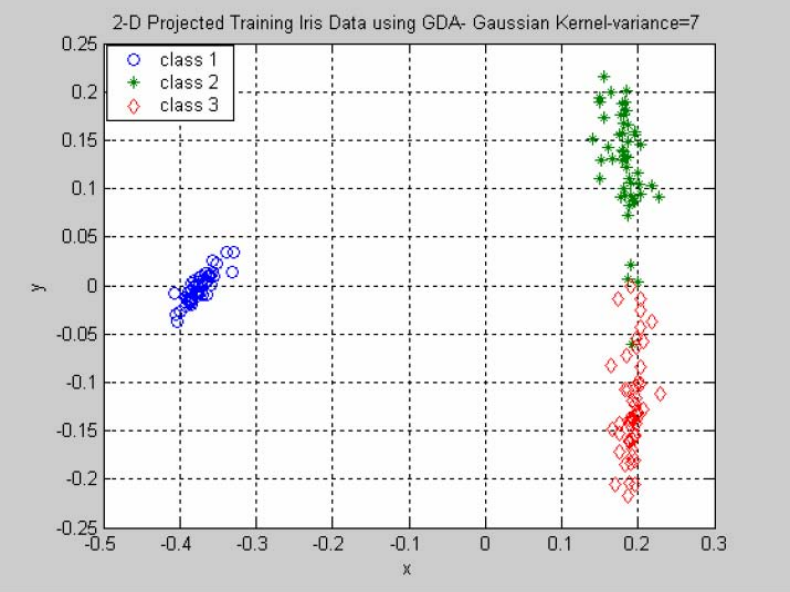

Figure 3.19 GDA projection of Iris data set, large variation ($2\sigma^2=7$) (from [Domboulas, 2004]).

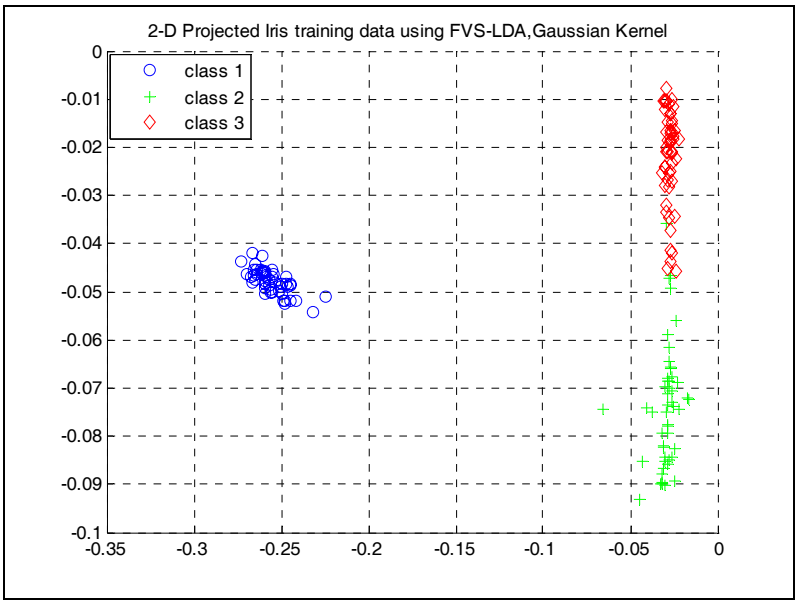

Figure 3.20 FVS-LDA projection of Iris data set, large variation ($2\sigma^2=7$), 15 FVs selected (max 150 FVs).

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

This chapter illustrates the application of the Bayesian Classifier presented in Chapter II and the FVS-LDA approach proposed by Baudat and Anouar to the IR face database used in our study. First, we discuss the distances combined with the learning algorithms and the types of the kernels considered for the nonlinear FVS-LDA classifier. We are interested specifically in determining the effect of combining different distance metrics with the available kernel functions while varying the kernels' parameters in the classification performance. The kernel parameters' tuning is obtained via a cross-validation scheme that we will discuss in further detail.

Next, we introduce some commonly used performance measures for evaluating biometrics systems, namely the Cumulative Matching Characteristic (CMC), or Rank Score, and the Receiver, or Relative Operating Characteristic (ROC), curve. We also present the confidence interval, a measure of the reliability of the obtained averaged results. Finally, we review *the 5x2cv paired t-test* introduced by [Dietterich, 1998] to evaluate whether the GDA and FVS-LDA approaches lead to statistically different classification results. Finally, we apply these concepts to the IR database and present results obtained.

A. THEORETICAL BACKGROUND

1. Introduction

This section first presents the nearest centroid-classifier approach, and the various distance metrics considered in our implementation of the FVS-LDA approach. Second, two types of experiments are discussed for evaluating pattern classification schemes, an identification and a verification experiment. Next, the confidence interval concept is reviewed and issues dealing with tuning of the nonlinear kernel function via cross-validation are introduced. Finally, a *5x2cv paired t-test* is applied to compare classification algorithms performances.

2. The Nearest Centroid Classifiers and Distance Metrics

The type of classifiers considered in our study is derived from nearest-centroid classification schemes. To classify an unlabeled pattern we consider its "distance" to all the class centroids estimated based on the training set. The pattern is assigned to the class

associated with the closest distance to a class centroid. The centroid of class C_j is defined as

$$\boldsymbol{m}_{C_j} = \frac{1}{|C_j|} \sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i, \tag{4.1}$$

where $\boldsymbol{x}_i$ s are the training samples belonging to class C_j. For any unlabeled pattern $\boldsymbol{x}$ the classification rule assigns a label according to the condition

$$C(\boldsymbol{x}) = \arg\min_{C_j} d(\boldsymbol{m}_{C_j}, \boldsymbol{x}), \tag{4.2}$$

where $d(\boldsymbol{x}, \boldsymbol{y})$ is a distance metric.

There are many different metrics reported in the literature to express the distance between two objects [Socolinsky, 2003]. Therefore, the importance of selecting an appropriate metric must be emphasized. In practice, a reasonable approach is to test the most commonly used metrics and evaluate them for specific data. In regard to our study, three different dissimilarity (i.e., distance) measures are defined below:

Euclidean Norm 2: $$L^2(x, y) = \sqrt{\sum_{i=1}^{N} \left(x^i - y^i \right)^2}, \; x, y \in \mathbb{R}^{N \times 1}, \tag{4.3}$$

Euclidean Angle: $$ang(x, y) = \arccos \frac{\sum_{i=1}^{N} x^i y^i}{L^2(0, x) L^2(0, y)}, \tag{4.4}$$

Mahalanobis Angle: $$M^{ang}(x, y) = \arccos \frac{\sum_{i=1}^{N} \sigma_i^{-1} x^i y^i}{L^2(0, x) L^2(0, y)}, \tag{4.5}$$

where x^i and y^i are the i^{th} coordinates of the x, y vectors and σ_i^{-1} is the inverse of the data variance estimate along the i^{th} axis. The Euclidean norm 2 is usually preferred over others due to its simplicity and good behavior, while the Euclidean Angle and the Mahalanobis Angle are preferred because they take values between [0, π] so they can be normalized to fall between [0, 1], which is needed for a verification type of test.

3. Classifier Performance Measures

Designing a pattern classification system involves several tasks: data preprocessing, feature extraction and selection, learning algorithm design, and, finally, the classifier performance evaluation task. The latter is required not only for a meaningful comparison between different classification schemes but also to measure their strengths and limitations when applied to complex scenarios. A significant amount of research has

been conducted toward that objective; in fact, the American National Standards Institute (ANSI) has approved specific standards that deal with the issues of required test sizes, performance statistics, error reporting, and presentation of performance results.

At this point, we need to introduce some concepts related to biometric identification technology, such as IR-image face recognition, before reviewing the classifier performance measures considered in our study. At first glance, biometric terms such as *recognition*, *identification*, and *verification* may seem identical, but this is incorrect. Each term denotes a different case:

- *Verification* is the task of either approving or rejecting a person's claimed identity (i.e., the label of a testing sample) based on a comparison of the collected sample with a previously registered sample (i.e., a training sample).
- *Identification* is the task of determining the identity of an individual. In these cases, the collected sample is compared to all the available patterns in the database (i.e., the training data). Our study is limited to "closed set" cases, in which all the possible tested individuals (i.e., classes) are included in the database. In other words, an answer will be given as the subject exists in the database.
- *Recognition* is the awareness that something perceived has been perceived before. "It is a generic term and does not necessarily imply either verification or identification."

It is clear that the three terms are not interchangeable in classification tasks. In the following section we will present two different types of experiments carried out to evaluate the biometrics system's performance along with the associated performance measures.

a. Identification Experiment: The Cumulative Matching Characteristic (CMC)

The Cumulative Matching Characteristic measure refers to a so-called closed-set identification experiment according to which every class in the testing data is also included in the training data set. To be consistent with the terminology used in the literature regarding data separation, we will define the subsets of the available data, namely, the *gallery* G and the *probe* P_G sets [Grother, 2003]. For an identification type of experiment, *gallery* G contains the estimated class centroids representing the set of

classes (i.e., individuals) collected in our database. Similarly, the *probe* P_G set contains the testing images, such that every image has a "match" in the gallery.

Next, each of the testing images is presented to the recognition system. Thus, a matrix of similarity measures s_{ij} is obtained, where i refers to the patterns stored in the data base, and j to the associated testing pattern. Note that in our study the similarity measures are considered as the distances between the estimated class centroids and the testing samples, unlike other studies in which images are compared to images directly [Grother, 2003]. In the following section, the idea of the rank of the match is defined as introduced in [Grother, 2003].

For the set of the *C*-obtained class centroids $g_k, k = 1,...,C,$ for each of the probe samples $p_j, j = 1,...,$ *number of testing samples*, given that we know where the correct match occurs, the rank of the match is defined [Grother, 2003] as

$$rank(p_j) = \left| \left\{ g_k : s_{kj} \geq s_{ij}, \text{given that } id(g_i) = id(p_j) \right\} \right|, \forall g_k. \tag{4.6}$$

The indices *i* and *k* are used to subscript class-centroid elements, and index j to subscript the probe samples; $|\{.\}|$ denotes how many times the condition within the brackets is met for every g_k. The resulting similarity matrix has elements corresponding to the i^{th} row and the j^{th} column. Identification performance is then regarded as part of the probe samples whose rank of match is *r* or lower, that is,

$$C(r) = \{ p_j : rank(p_j) \leq r \}, \forall p_j. \tag{4.7}$$

Thus, the Cumulative Match Score as a function of rank *r* may be defined as

$$P(r) = \frac{|C(r)|}{|P_G|}. \tag{4.8}$$

Now we need to know how likely it is that the top similarity score occurs at the position of the correct match, which is referred to as the rank-1 score. Accordingly, rank-2 is defined as the probability of assigning the correct class to a testing label within the top two choices of the assignment step. Generalizing the concept, we can state that

rank-N is the probability of correctly assigning a test sample in the classes associated to the N top similarity scores. A typical CMC curve is depicted in Figure 4.1 for a case of a 140-individual (i.e., classes) database.

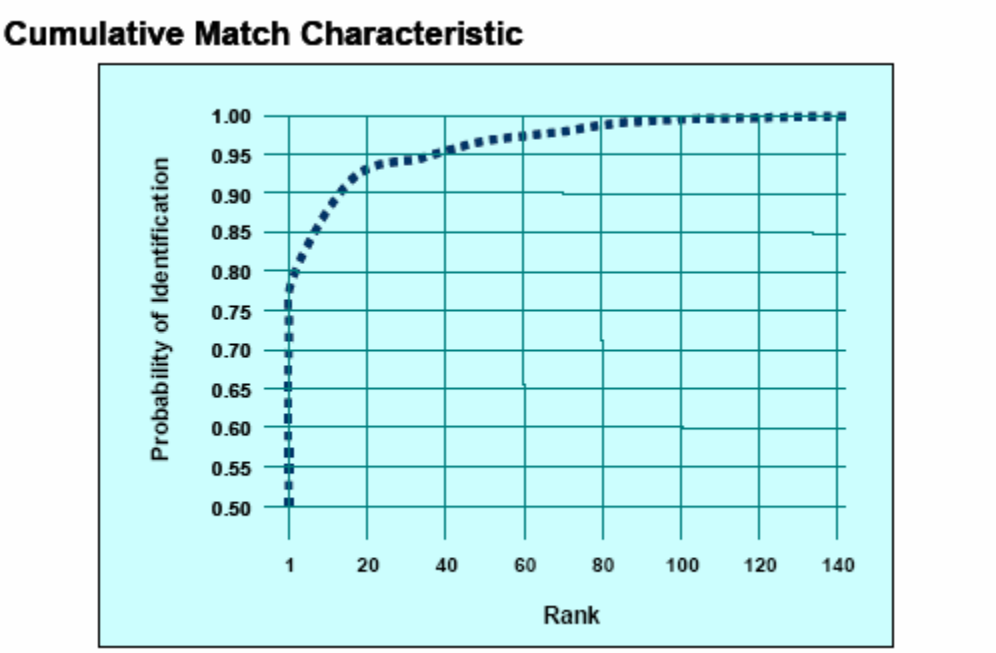

Figure 4.1 Example of the Cumulative Match Characteristic Curve (from [Biometrics, 2006]).

An interesting feature of the CMC curve is that the probability of identification at rank-1 is a widely used overall classification-rate measure defined as the ratio of the number of patterns correctly classified over the total available number of testing patterns. Moreover, the probability of identification is 1 at the highest rank in a closed set experiment. Thus it is important to report the number of classes included in the database when referring to rank scores [Biometrics, 2006].

b. Verification Experiment: The Receivers Operating Characteristic (ROC)

Unlike the identification task, verification may be viewed by some as a more realistic type of experiment for application purposes. In practice, this test addresses the case of an individual's claimed identity (i.e., class label) at a checkpoint. From the biometrics collected at the time, a decision should be made whether the claim is true or false. That task is discussed next following the analysis in [Grother, 2003].

Note that for the verification type of experiment the so-called *imposter* set P_N needs to be defined. The imposter set P_N contains the testing images that do not have a "match" in the gallery, and is used to simulate possible intruders in the system. Like the identification type of experiment, we compute the same similarity matrix of entries s_{ij}, and our decision is based on a selected threshold *t*, varying between [0, 1], which first requires normalizing the similarity measures selected for the test, provided they are bounded.

For any given threshold *t*, two types of errors may occur: a "false reject in which the system incorrectly matches the probes below threshold" or a "false accept in which an imposter claims an identity and is matched by the system above threshold" [Grother, 2003]. A system with a low probability of the false acceptance rate P_{FA} and a low probability of the false rejection P_{FR} is desirable; however, this is not feasible. To see the qualitative behavior of the two types of error, the ROC curve is used.

The ROC curve is the plot of probability of the false acceptance P_{FA} against the probability of correct identification (or correct verification) P_{CI} ($P_{CI} = 1 - P_{FR}$), where

$$P_{FA}(t) = \frac{\left|\left\{s_{ij} : s_{ij} \geq t\right\}\right|}{\left|P_N\right| + \left|G\right|}, \forall p_j \in P_N \tag{4.9}$$

and

$$P_{CI}(t) = \frac{\left|\left\{p_j : s_{ij} \geq t, \text{given that } id(g_i) = id(p_j)\right\}\right|}{\left|P_G\right|}, \forall p_j \in P_G. \tag{4.10}$$

A visualization of that trade-off relationship is depicted in Figure 4.2 below, where each operating point in the curve refers to a different threshold value *t*. The area under the ROC curve is indicative of the system's performance; in fact, a large area implies a well-designed system.

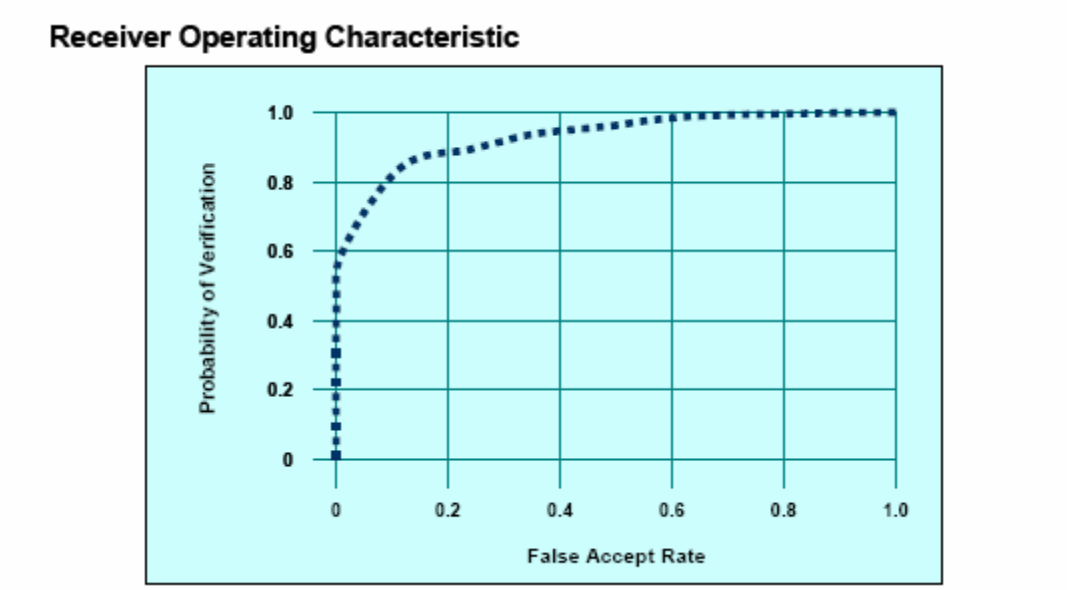

Figure 4.2 Example of the Receiver Operating Characteristic curve (from [Biometrics, 2006]).

4. Confidence Interval

Merely providing a single value for the value of an unknown parameter of a specific population is not that useful. Obviously, the range of values for the estimated parameter is related to the available sample. Thus, reporting an estimate for that parameter should always be in conjunction with the associated *confidence interval (CI)*.

There are alternative CI interpretations; a common one is as follows. For a given CI (e.g., 95%), "the 95% CI is the interval in which we are 95% certain contains the true population value as it might be estimated from a much larger study" [Camcode, 2006]. Note that the value under investigation is not restricted to the mean, and CIs can be computed for the median, the difference between two means, a proportion, etc. [StatsDirect Ltd, 2006].

The mechanism for computing the CI when the observed data are not from a normal distribution is the percentile method [StatsDirect Ltd, 2006]. In our case, the classification rates obtained for over 1,000 iterations are not normally distributed (see Figure 4.3 below) and the percentile method is used to compute the confidence interval for the classification rate. We proceed as follows to compute the 95% confidence level obtained from a set of 1,000 iterations:

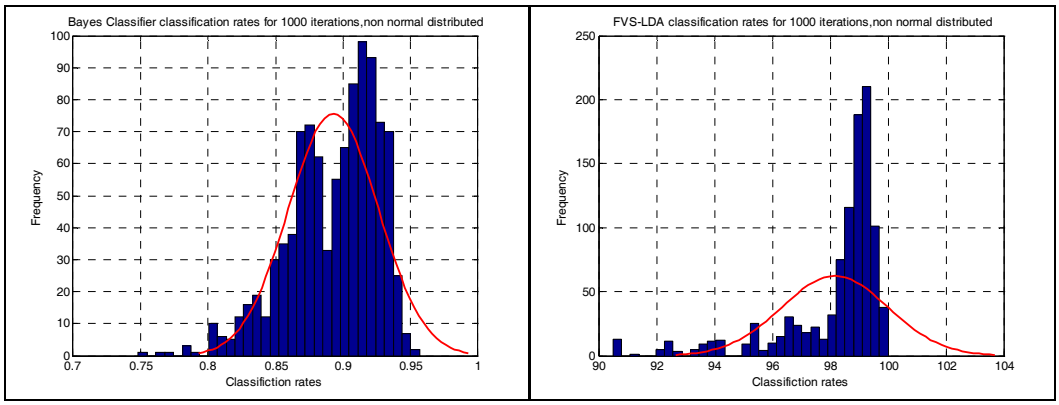

Figure 4.3 Classification rates for Bayes and FVS-LDA histogram classifiers. Populations are non-normally distributed.

- Arrange in increasing order the 1,000 values for the classification rate.
- Find the values related to the endpoints of a 95% CI (i.e., 2.5*1,000 and 97.5*1,000).
- The endpoints of the interval are the 25th and the 975th values.

5. Kernel Selection

The selection of a suitable type of kernel function remains a challenging issue for nonlinear kernel-based classifiers. Moreover, setting up the parameters associated with the type of kernel function is related to the nature of the observed data and is usually time-consuming and computationally expensive. Next, we list the kernel functions used in our study and present a widely applied technique for parameter tuning, the cross-validation method.

a. Kernel Function Types Considered

There are numerous types of kernel functions that may be used, depending on the data characteristics. In our study, the following types of kernel functions were considered:

the Gaussian type: $$k(x,y)=\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right),\ \text{where}\ \sigma>0,\ \text{and} \tag{4.11}$$

the Polynomial type: $$k(x,y)=\langle x,y\rangle^2,\qquad k(x,y)=(\langle x,y\rangle+1)^2. \tag{4.12}$$

b. Parameters Selection: The Cross-Validation Scheme

In practice, kernel parameters, that is, the polynomial degree or the variance of the Gaussian kernel, are selected using a *K*-fold cross-validation scheme. This

process involves, first, partitioning the data into *K* segments of equal size. Next, we perform *K* training runs, in which the training in each run is based on *K-1* segments and test on the one left out. We compute the classification error rate for each run and average over the total *K* runs. Finally, we choose the parameters associated to the smallest error rate. For a data set of size *n*, the extreme case of *K* is *K*=*n*, known as the leave-one-out method. A typical choice of *K*=10 or *K*=5 is adequate to perform the cross-validation scheme [Hastie, 2001].

6. The 5x2cv Paired *t*-Test

The 5x2cv paired *t*-test is a statistical test applied to compare classification learning algorithms. In other words, we seek an answer to the following question posed by [Dietterich, 1998]: "Given two learning algorithms A and B and a small data set S, which algorithm will produce a more accurate classifier when training on a data set of the same size as S?" Before we can discuss the test in more detail, we need to introduce the terminology related to the types of tests that are designed to check whether one learning algorithm outperforms another in a particular learning task, as presented in [Dietterich, 1998]:

- *Type I error:* The probability of incorrectly detecting a difference when no difference exists.
- *Type II error:* The failure to detect a real difference between algorithms.
- *Null hypothesis to be tested:* For a randomly drawn training set *R* of fixed size the two learning algorithms will have the same error rate on a test example randomly-drawn from an available data set *X*.

The 5x2cv paired *t* test to be described next was proposed by Dietterich [1998]. In fact, this test is a modification of the *k*-fold cross-validated paired *t*-test discussed also in [Dietterich, 1998], including 5 replications of a 2-fold cross-validation. First, the available data set *S* is separated into a training set S_1 and a test set S_2 of equal size. Next, both A and B learning algorithms are trained on the training set; thus, classification schemes $\hat{f}_A$ and $\hat{f}_B$ are obtained. For any given testing sample $x \in S_2$, we keep track of whether or not it has been correctly classified. The results may be tabulated in the matrix format depicted below in Table 4.1:

Number of testing samples misclassified by both $\hat{f}_A$ and $\hat{f}_B$, denoted as $\boldsymbol{n}_{00}$	Number of testing samples misclassified by $\hat{f}_A$ but not by $\hat{f}_B$, denoted as $\boldsymbol{n}_{01}$
Number of testing samples misclassified by $\hat{f}_B$ but not by $\hat{f}_A$, denoted as $\boldsymbol{n}_{10}$	Number of testing samples misclassified by neither $\hat{f}_A$ nor $\hat{f}_B$, denoted as $\boldsymbol{n}_{11}$

Table 4.1 McNermar's contingency table (after [Dietterich, 1998]).

Note that the total number of training samples is $\boldsymbol{n}_{00} + \boldsymbol{n}_{01} + \boldsymbol{n}_{10} + \boldsymbol{n}_{11} = \boldsymbol{n}$. That process produces the following error estimates: $p_A^{(1)}$ and $p_B^{(1)}$, when trained on S_1 and tested on S_2, while $p_A^{(2)}$ and $p_B^{(2)}$, when trained on S_2 and tested on S_1. The terms p_A and p_B are the proportions of the test examples incorrectly misclassified by algorithms *A* and *B* respectively defined

$$\begin{aligned} p_A &= (\boldsymbol{n}_{00} + \boldsymbol{n}_{01}) / \boldsymbol{n} \\ p_B &= (\boldsymbol{n}_{00} + \boldsymbol{n}_{10}) / \boldsymbol{n}. \end{aligned} \tag{4.13}$$

We obtain two estimated differences, $p^{(1)} = p_A^{(1)} - p_B^{(1)}$ and $p^{(2)} = p_A^{(2)} - p_B^{(2)}$. Finally, we compute the estimated variance

$$s^2 = (p^{(1)} - \overline{p})^2 + (p^{(2)} - \overline{p})^2, \tag{4.14}$$

where $\overline{p} = (p^{(1)} + p^{(2)}) / 2$. The following statistic can be defined, based on the previous derivation,

$$\tilde{t} = \frac{p_1^{(1)}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s_i^2}}, \tag{4.15}$$

which is called the 5x2cv $\tilde{t}$ statistic. The term $p_1^{(1)}$ in the numerator refers to the estimated $p^{(1)}$ from the first replication, while s_i^2 denotes the variance of the i^{th} replication. According to [Dietterich, 1998], "$\tilde{t}$ has approximately a *t* distribution with five degrees of freedom." Thus the mechanism behind the 5x2cv test is to compare the distribution of the latter ratio based on the observed counts to the one based on the assumption that the null hypothesis is true. In fact, it comes down to comparing the ratio based on the observed counts to the argument of a *t* distribution with five degrees of

freedom for a given level of significance. The null hypothesis is rejected when the estimated ratio is greater than the threshold, meaning the two algorithms have different performances.

Note that the experimental results using the 5x2cv test reported in [Dietterich, 1998] show that this is a low Type I error test. Moreover, the 5x2cv test is a powerful (i.e., low Type II error) when the task is to be confident that there is no observed performance difference. Unlike other candidate tests, the 5x2cv test also considers the effect of varying the training sets by repeating the training and testing process for different data separations.

B. EXPERIMENT DESCRIPTION

1. Database Information

Our experiment used the uncooled IR image face database collected in an earlier study [Lee, 2004]. Images are 8-bit/pixel with (160×120) pixels spatially and 60mK temperature resolutions. The database includes frontal views of 50 adult subjects (i.e., classes). Variations in angle and tilt were introduced in the collected photographs, since each one started at one of ten fixed points behind the camera. Further variations in facial expression were introduced by considering the following three poses for each class:

- a neutral pose,
- a pronouncing-the-vowel-"u" pose, and
- a smiling pose.

All subjects participated only once (i.e., all images per class were collected the same day). Further variations due to different lightning conditions were not considered. In addition, each subject was sampled ten times for each of the three poses, resulting in 1,500 grey-scale images. Next, the images were cropped down to (60×45) pixels to retain the middle section of the face only and were reshaped into column vectors of size (2700×1). The overall data matrix used was of dimension (2700×1500).

In face-recognition experiments, the following image sets are formed:

- A *training set* is used to form the feature space for pattern representations, such that class centroids are estimated.
- A *gallery set G* according to [Grother, 2003] "contains identically one signature per subject and represents the set of images that have been

enrolled in the system." However, since for comparisons in our experiments we used the distances of an image from class centers, the gallery set include the class centroids as a subject signature.

- A *testing set* contains the images that will evaluate the system's performance. More specifically, the testing set is called *probe set* P_G and represents a legitimate user when an identification closed-universe experiment is performed (i.e., when each of the testing images has a match in the gallery), [Grother, 2003]. On the other hand, the term *imposter set* P_N is used for the testing data in an open-universe verification experiment (i.e., when not all of the testing images have a match in the gallery) to represent individuals attempting to defeat the system [Grother, 2003].

For benchmarking purposes we followed the same percentage data separation considered in a previous study [Domboulas, 2004], a 60–40% partitioning for training and testing data, respectively. Similarly, 1,000 iterations were considered to count for the random variations in the selection of the training and testing data.

2. Software Implementation

The main experiments carried out in this study are listed below:

- Bayesian Classifier algorithm approach to compare results with linear classifiers considered in previous studies,
- Kernel-function parameters tuning using the cross-validation scheme,
- Implementation of the FVS-LDA algorithm to compare results to the GDA approach considered in [Domboulas, 2004],
- Evaluation of the classification schemes for the identification and the verification experiment using the Cumulative Match Characteristic curve, along with the associated confidence intervals and the Receiver Operating Characteristic curve, respectively, and
- Application of the 5x2cv paired *t*-test to compare the FVS-LDA and GDA classification performances based on experimental results.

The software code developed in this study is included in Appendix B.

3. Bayes Classifier Results

Recall that the underlying theory of the Bayesian scheme investigated here is based on an estimation of the multivariate density function of the *l*-dimensional training patterns. For the data separation considered in our study there are 18 training patterns per class and the maximum number of features equals 17 (i.e., l=17). This characteristic is due to the fact that the features used to describe the data are derived from a singular value decomposition of a per-class covariance matrix of size 18 by 18 based on the training

data, which is of rank 17 (see Chapter II). In general, the rank of the class covariance matrix depends only on the number of training samples per class. Next, we implemented the D'Agostino-Pearson goodness-of-fit test to evaluate whether the data under investigation deviates from multivariate normality or not.

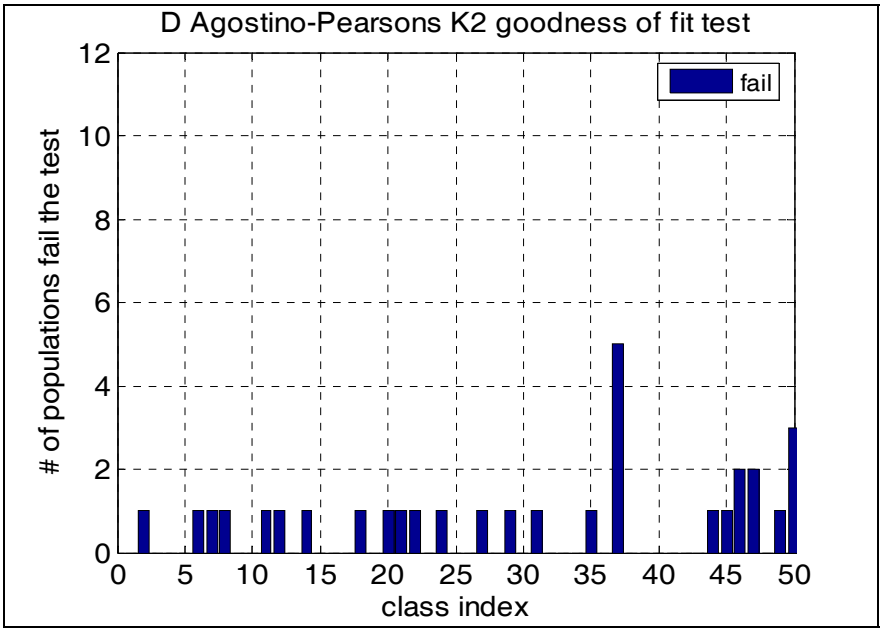

Figure 4.4 Goodness-of-fit test, marginal densities per class considered.

For a given class, the images are projected onto a feature space of dimensionality up to the rank of the per-class covariance matrix (i.e., 17). Next, given that each class is represented by 18 (i.e., 60%) training samples, the resulting per-class data matrix is of size (17 x 18). The goodness-of-fit test is performed along each row of the obtained matrix. Recall that marginal normality does not imply multivariate normality [Gnanadesikan, 1977]. Results shown in Figure 4.4 indicate that a significant number of classes fail the marginal normality test on one of their features. For example, one feature of class 1, 6, 8, 9, fail the normality test, while five features of class 37 fail the normality test for a level of significance equal to 0.05. Even though results indicate the data deviate from multivariate normality we still applied the Bayes classifier modeling the data as multivariate Gaussian as it was simple to implement to provide a baseline performance.

Following, we present some points concerning the Bayes Classifier implementation that are of interest.

- Selecting the optimum number of features is affected by the mechanism described next. Increasing the number of features (i.e., singular values) increases the classification performance up to a point beyond which the classification rates start to decrease. That effect is referred to in the literature as the "peaking effect" [Lu, 2003]. In our study, the number of training patterns per class determines the maximum number of features equal to 17 for the Bayes Classifier. Figure 4.5 illustrates the "peaking effect." Note that the per-class covariance matrices become singular when 18 singular values are used; thus the classification ratio becomes zero.

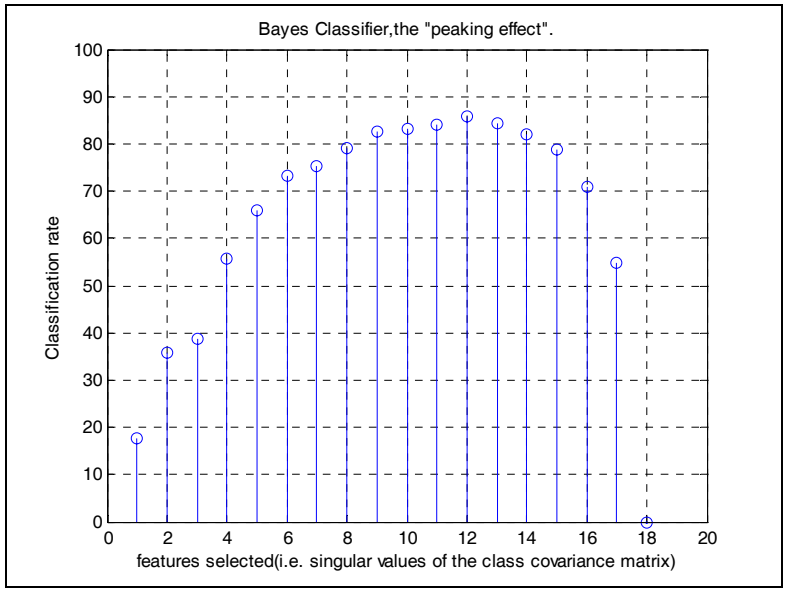

Figure 4.5 Bayes Classifier: the "peaking effect".

- Note also that the optimum number of features varies for each iteration affected by the training and testing data separations. The histogram below shows the frequency at which the optimum number of features occurs between nine and fourteen over 1,000 iterations. Obviously, selecting a fixed value of eleven features is better suited.

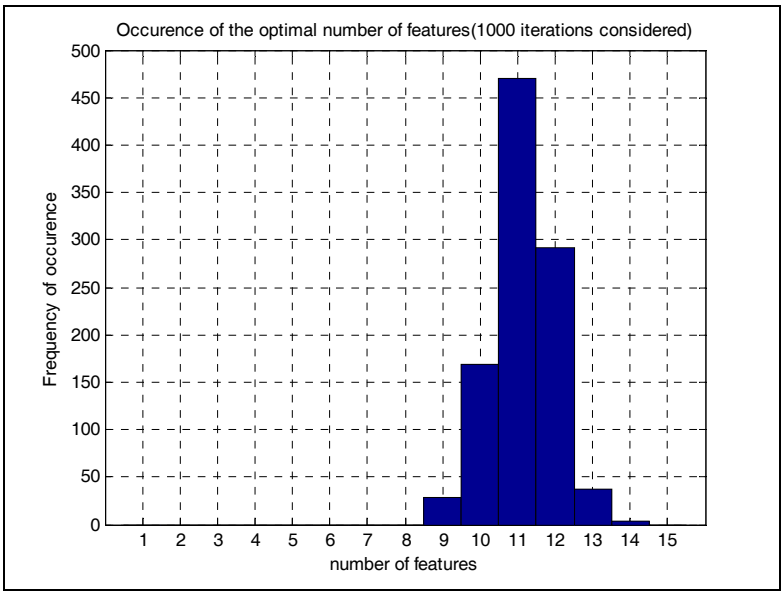

Figure 4.6 Bayes Classifier: optimal number of features.

Considering those two points, we implemented a scheme in which we selected a fixed number of features to 11. Results are depicted below:

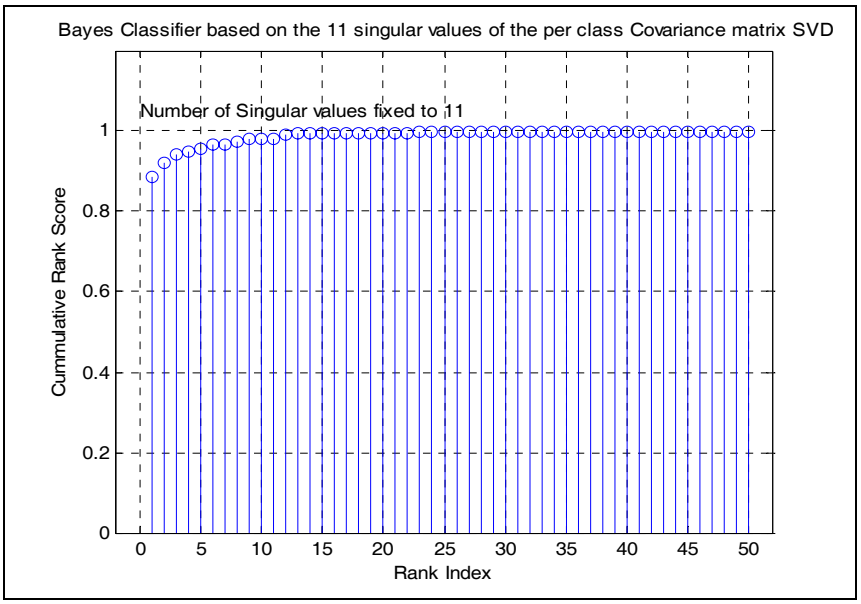

Figure 4.7 Bayes Classifier Cummulative Rank Score for 11 singular values.

Moreover, the Bayes Classifier with 15 singular values was implemented for comparison to the PCA with 15 eigenvalues selected (referred to as PCA15), as previously studied by [Lee, 2004]. Results are depicted in Figure 4.8 in terms of the Cumulative Rank Score curves.

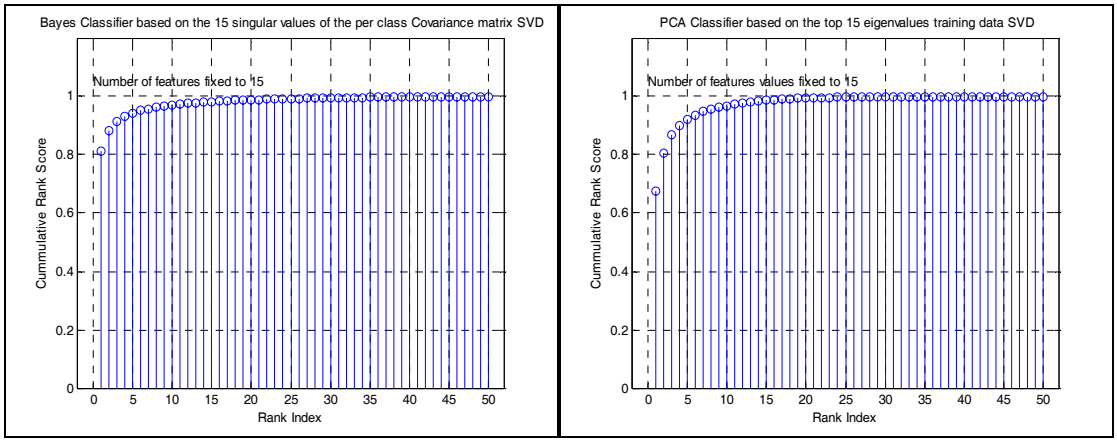

Figure 4.8 Bayes Classifier compared to PCA when 15 features are selected.

Note that, the Bayes classifier outperforms the PCA scheme for such a low-dimensional image representation. Moreover, the comparison to the PCA15 is even superior when fewer features are selected, since in most of the cases the Bayes classifier attains its maximum performance for eleven to twelve features. The optimum performance for the Bayes classifier based on a varying-features basis and the per-class classification performance are plotted in Figures 4.9 and 4.10.

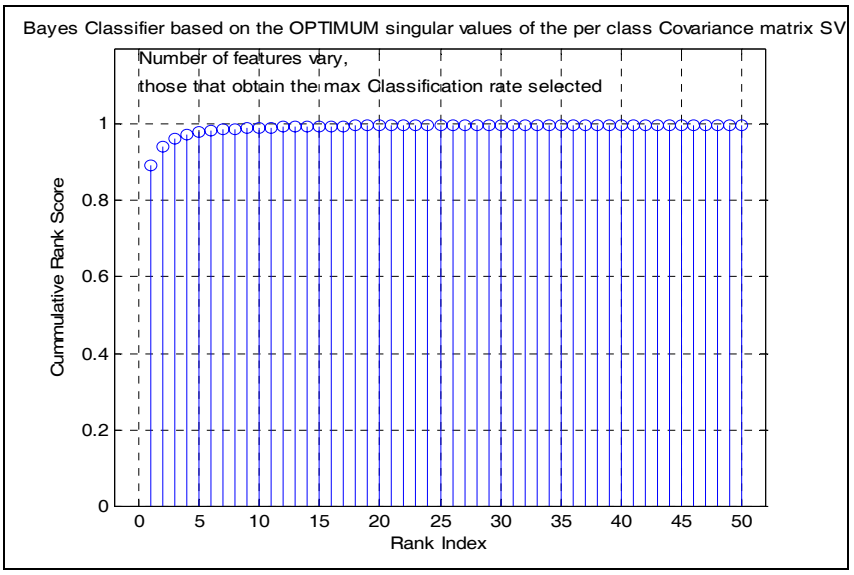

Figure 4.9 Bayes classifier based on an optimum features selection.

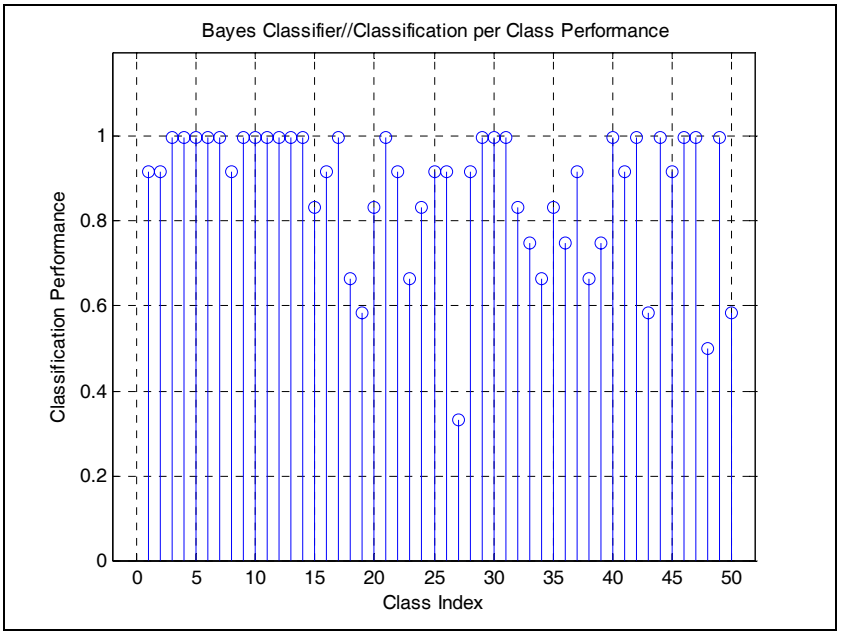

Figure 4.10 Per-class classification performance of the Bayes classifier.

Rank-1 results for the Bayes Classifier are tabulated below, along with the associated confidence intervals.

Rank-1 Scores/Confidence Intervals for a 95% confidence level, 1000 iterations considered			
Classification scheme	Features selected (singular vectors/eigen vectors)		
	15	11	optimum
Bayes Classifier	82.40 [72.5 88.00]	90.16 [80.00 94.00]	90.56 [80.00-94.67]
PCA	68.57 [61.33-73.17]	x	x

Table 4.2 Bayes Classifier performance compared to PCA.

4. FVS-LDA Distance and Kernel Parameters' Impact on Classification Results

The FVS-LDA classification scheme considered in this study derives from feature selection schemes. Therefore, the selection of the number of features not only has an impact on the classification performance but also is associated with the computational cost, provided the kernel function parameters are well tuned. For that purpose, the first important step in the FVS-LDA algorithm was to select the variance for the Gaussian kernel function and the order of the polynomial kernel.

The values for those parameters were obtained using the *K*-fold cross-validation scheme for K=5 segments, setting the number of feature vectors (FVs) equal to 350 (the maximum number of FVs is equal to the number of the available training data, i.e., 900) while varying the variance from the order of 10^6 to 10^8. However, the choice of a specific distance metric remains a crucial issue and should not be ignored, thus the kernel parameter tuning was performed in conjunction with the distances considered in our study. Results are tabulated below. Note also that once the kernel parameters have been selected, they remain the same when different numbers of FVs are considered.

K-fold cross-validation($K=5$)	Euclidean Norm	Mahalanobis Angle	Euclidean Angle
Feature Vectors selected (FVs=350)			
Gaussian Kernel	variance=$1.4*10^7$	variance=$2.6*10^7$	variance=$1.8*10^7$
Polynomial Kernel	order =2	order=2	order=2

Table 4.3 Kernel parameters selection using a cross-validation scheme.

We selected four different numbers of feature vectors, namely, 250, 350, and 450, and 600 FVs. The purpose is to compare the FVS-LDA algorithm performance with the previous study conducted by [Domboulas, 2004] on the GDA approach for a similar size of the associated kernel matrices. Note that although the complexity of the problem is related to the number of the FVs or to the eigenvalues of the kernel matrix for the FVS-LDA and GDA, respectively, the mechanism behind features' extraction is different. Thus those terms should not be used interchangeably.

5. Identification Experiment

The data separation considered for the identification experiment was the same as was followed in the Bayes classifier, namely, 60% of the data used for training (i.e., 900 patterns) and the remaining 40% used for testing (i.e., 600 patterns). The performance metrics considered for use for that type of experiment is the Cumulative Rank Characteristic curve (Rank-one score). Rank-one scores (i.e., Average Recognition Rates) and the associated confidence intervals are tabulated in Table 4.4.

Kernel functions	FVs	Average Recognition Rate (%); [95 % Confidence interval]					
		Distances					
		Euclidean Norm [Gaussian kernel variance=$1.4*10^7$]		Mahalanobis Angle [Gaussian kernel variance=$2.6*10^7$]		Euclidean Angle [Gaussian kernel variance=$1.8*10^7$]	
Gaussian	250	96.66	[91.5, 98.83]	93.18	[85.66, 96.33]	96.35	[91, 98.5]
	350	97.75	[91.83, 99.33]	95.18	[89, 97.83]	97.57	[92.17, 99.33]
	450	**98.32**	**[92.83, 99.83]**	97.22	[91.17, 99.17]	98.16	[92.5, 99.83]
	600	98.50	[94, 99.83]	x	x	x	x
Polynomial	250	95.35	[88.33, 98]	95.04	[88.5, 97.33]	96.80	[92.17, 98.67]
	350	97.08	[91.16, 99.16]	97.26	[92.33, 98.83]	98.07	[93.83, 99.33]
	450	97.33	[89.83, 99.33]	**98.02**	**[93.33, 99.67]**	**98.39**	**[94, 99.67]**
	600	x	x	94.48	[94.17, 99.67]	98.48	[94.17, 99.67]

Table 4.4 FVS-LDA average recognition rates and 95% confidence intervals, 1,000 iterations.

6. Verification Experiment

The verification type of experiment requires constructing an *imposter* set P_N. Recall that the imposter set P_N is defined as containing the testing images that do not have a "match" in the gallery and is used to simulate possible intruders in the system. Following is the data separation considered for that purpose. The training set contains 30 classes with 18 patterns each, resulting in a total of $540\,(18\times 30 = 540)$ training patterns

used to estimate the class centroids. The probe set includes the remaining patterns of the same 30 classes, namely, 12 patterns per class, resulting in a total of $360\,(12\times 30=360)$ used for the correct identification process. Thus, what is left is considered the imposter set, consisting of 20 classes each of 30 patterns, resulting in a total of $600\,(20\times 30=600)$ patterns used to model the false-alarm rate of a system.

Next, we used the Receiver Operating Characteristic curve (ROC) and the associated equal error rate (EER) to evaluate the performance of the FVS-LDA scheme. The EER is defined as the intersection point between the probability of a false-alarm curve and the probability of incorrect rejection. Equivalently, the area under the ROC curve may be used to evaluate the performance of a system. A large area implies a system of high performance when the probability of false alarms is plotted versus the probability of correct identification. The ROC curve and the associated EER rates obtained for the FVS-LDA scheme, along with the rank-one scores for 100 iterations, are provided in Figures 4.11 and 4.12. The FVs selected were such that the percentage to the maximum possible number of features (i.e., FVsmax=540) is similar to the identification experiment. The distance metrics considered for this experiment were those that are bounded, such as the Mahalanobis angle and the Euclidean angle.

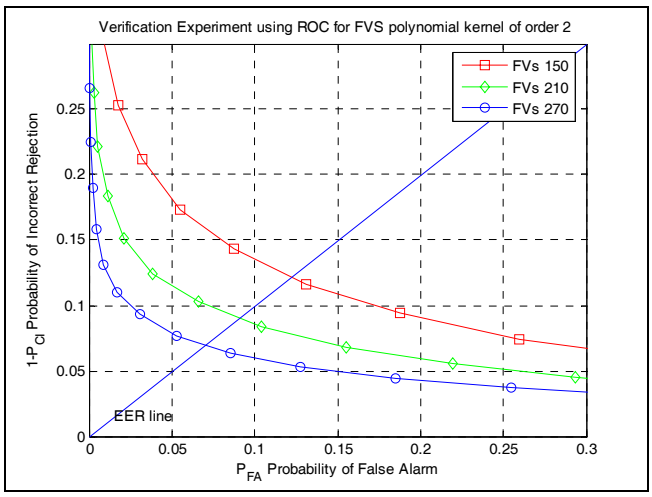

Figure 4.11 FVS-LDA verification performance for a second-order polynomial kernel combined with the Mahalanobis angle distance.

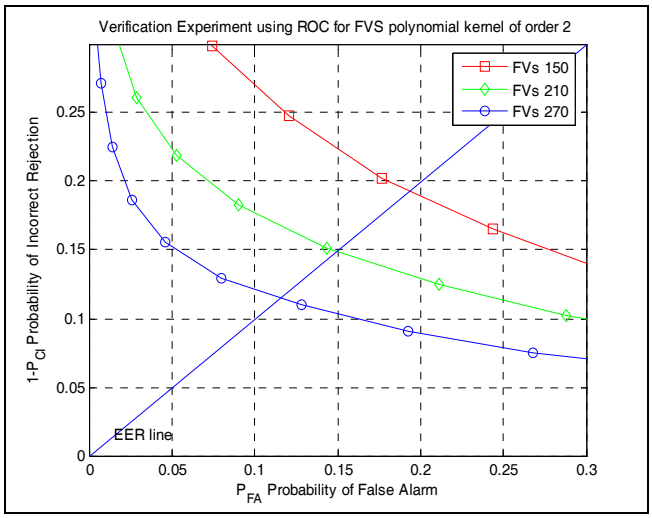

Figure 4.12 FVS-LDA verification performance for a second-order polynomial kernel combined with the Euclidean angle distance.

Algorithms	No of FVs	Distances			
		Mahalanobis Angle		Euclidean Angle	
		Recog Rate (%)	EER (%)	Recog Rate (%)	EER (%)
FVS-LDA	150	**97.56**	**12.2**	97.81	19.29
	210	**98.29**	**9.05**	98.12	14.9
	270	**98.58**	**7.01**	98.70	11.51

Table 4.5 Equal Error Rates(EER) for a verification experiment: 30 subjects in the gallery, 50 subjects in the probe, 100 iterations.

Results show that the FVS-LDA approach combined with the Mahalanobis angle distance outperforms that obtained with the Euclidean angle option for the type of verification scenario implemented. Next, we consider the same verification experiment to compare the GDA and FVS-LDA performances.

7. Comparing GDA and FVS-LDA Algorithms

The novel feature-selection algorithm FVS proposed by Baudat and Anouar [Baudat, 2003], combined with the classical LDA classification scheme, can approximate closely its "kernelized" version, namely, the GDA scheme previously considered in [Domboulas, 2004]. Moreover, the related computational time is significantly reduced, since the size of the associated kernel matrices is reduced from the number of the total training data (GDA) to only the portion of the selected training data (FVS) [Baudat,

2003]. This section first provides a comparison based on obtained experimental results from the identification experiment using the 5x2cv paired *t*-test. Then the verification experiment is considered, along with the ROC curves, for comparing classification schemes. Finally, the computational load in terms of computational time per iteration is noted.

a. The 5x2cv Paired t Test

The 5x2cv paired t-test is a "statistical test for determining whether one learning algorithm outperforms another on a particular learning task" [Dietterich, 1998], based on estimated classification rates. FVS-LDA based identification results obtained are tabulated in Table 4.4, while the GDA-based results are depicted in Table 4.6 from a previous study [Domboulas, 2004].

Kernel function	No of Kernel Matrix Eigenvectors selected	Recognition Rate (%); [95 % Confidence interval] Mahalanobis Angle Distance	
Polynomial of order 2	100	89.44	[86.5, 92]
	150	94.48	[92.17, 96.5]
	200	96.70	[94.83, 98.17]
	250	97.74	[96.33, 99]
	500	**98.55**	**[97.33, 99.5]**
	Default (700)	98.41	[97, 99.50]

Table 4.6 GDA average recognition rates and 95% confidence intervals, 1,000 iterations, after [Domboulas, 2004]).

Results show that, in both cases, performance improves when the number of features/eigenvectors is increased, as expected. In fact, GDA outperforms FVS, and results become very close to each other only when almost half of the features/eigenvectors are considered. Thus we performed the 5x2cv paired t-test to check whether differences in the classification performances are statistically significant as the associated number of features/eigenvectors varies.

To this end, the available data set (i.e., 1,500 images) was separated into two equal-sized sets (i.e., each of 750 images); and the test was repeated for a variation of features/eigenvectors, from 50 to 600, in increments of 50. Since we are interested in GDA and FVS-LDA's relative performance, we selected the polynomial kernel function of second order for both schemes and the Mahalanobis angle distance, respectively.

Recall that this statistical test estimates the variable $\tilde{t}$, which is approximated to a t distribution with five degrees of freedom, provided the null hypothesis is true (i.e., the algorithms perform the same). Results are depicted below in Figure 4.13. Note that the threshold for a *t* distribution of five degrees of freedom is equal to $t_{threshold} = \pm 2.01505$ for a level of significance equal to 0.05. Thus the null hypothesis is rejected when $\tilde{t} > +2.01505$ or $\tilde{t} < -2.01505$.

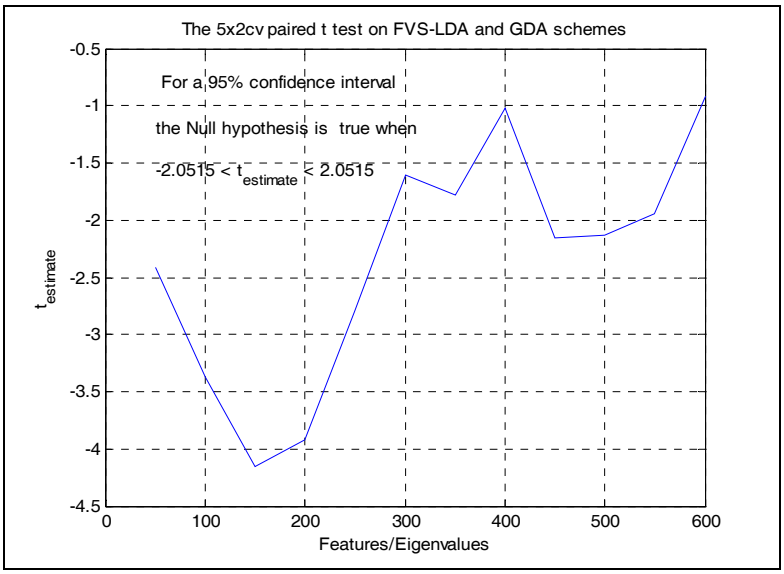

Figure 4.13 Comparison of the FVS-LDA and the GDA algorithms using the 5x2cv paired *t* test.

The fact that the training set size (i.e., 750 images) used for the 5x2cv paired *t* test is close to the separation used throughout our study for the identification experiments (i.e., 900 training samples) enables us to further generalize the results of the paired *t* test. Thus the following comments can be made on the results:

- GDA outperforms FVS-LDA in all cases. That is reflected in the negative sign of the obtained statistic $\tilde{t}$. Recall that the numerator in Eq. 4.15 is the estimated difference between the GDA and FVS-LDA error rates for the first replication; thus the negative sign implies a lesser error rate for GDA. That is consistent with the results reported in Tables 4 and 5 (a polynomial kernel of a second-order Mahalanobis angle distance for GDA versus the Euclidean angle for FVS-LDA).

- Selecting the number of features for the FVS-LDA scheme from a third to a half of the available features results in close to the standard GDA performance (i.e., the null hypothesis is considered to be true for a 0.05 level of significance). That is consistent with the suggestions made by Gaston Baudat (in a personal communication) on selecting the number of features used in the FVS algorithm.
- Further increasing the number of features results in differences between the algorithms that are not statistically significant (i.e., the null hypothesis is considered to be true).

b. Verification Experiment

For comparison purposes, we also considered the verification experiment. To evaluate the performance of the GDA and the FVS-LDA schemes, we used the Receivers Operating Characteristic curve (ROC) and the associated equal error rate (EER). The data separation follows the same partitioning as was introduced in the FVS-LDA performance evaluation using ROC curves. The associated ROC curve for the GDA algorithm is depicted in Figure 4.14.

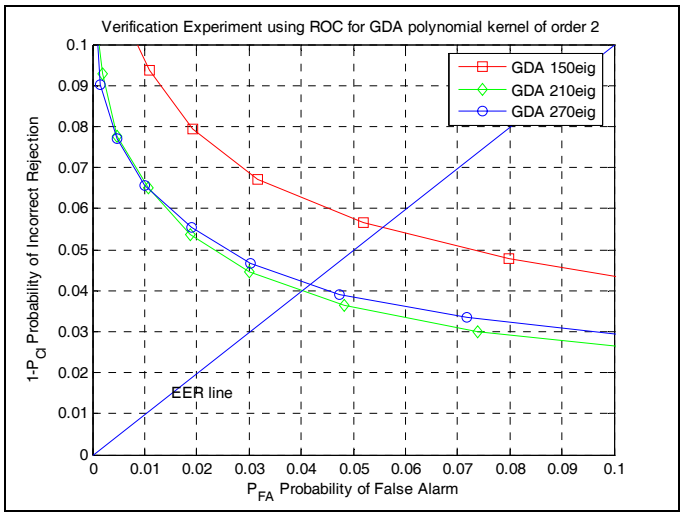

Figure 4.14 Verification experiment for the GDA approach; 100 experiments; Mahalanobis angle distance.

Table 4.7 presents results used to compare the ROC curves of Figures 4.11, 4.12, and 4.14 in terms of the EER.

Algorithms	No of Eigenvectors/ FVs (max 540)	Distances			
		Mahalanobis Angle		Euclidean Angle	
		Recog Rate (%)	EER (%)	Recog Rate (%)	EER (%)
GDA	150	98.471	**5.55**	---	---
	210	99.1	**4.01**	---	---
	270	98.90	**4.15**	---	---
FVS-LDA	150	97.56	**12.2**	97.81	**19.29**
	210	98.29	**9.05**	98.12	**14.9**
	270	98.58	**7.01**	98.70	**11.51**

Table 4.7 Equal Error Rates (EER) for the verification experiment: 30 subjects in gallery, 50 subjects in probe, 100 experiments.

Results for the verification experiment show that GDA outperforms the FVS-LDA in terms of the EER metric.

c. Computational Costs Differences

We can obtain an understanding of the associated computational cost for each of those schemes by considering the mean running time per iteration, given MATLAB was used for the implementation. Domboulas reported that, the GDA implementation with the Gaussian kernel-mapping function took about 7.5 to 8 minutes, while the implementation with a polynomial kernel of second order took about 75 seconds per iteration using a 3-GHz PC on the same database and data partitioning type as followed in our research [Domboulas, 2004]. For benchmarking purposes, we re-run the exact code utilized by Domboulas, [Domboulas, 2004] on the same machine used for our study to remove variations due to machine different characteristics. The implementation with a polynomial kernel of second order took about 45 to 50 seconds per iteration for a selection of eigenvectors equal to 250, 350 and 450. In our study, the running time per iteration implementing the FVS-LDA was significantly reduced. Results are listed in Table 4.8.

FVS-LDA						
	Gaussian kernel mapping			Polynomial second order kernel mapping		
Features	Euclidean	Mahalanobis angle	Euclidean angle	Euclidean	Mahalanobis angle	Euclidean angle
	mean running time per iteration					
250	8.90sec	10.0sec	9.30sec	9.70sec	13.6sec	10.0sec
350	15.2sec	16.6sec	16.0sec	16.7sec	22.7sec	17.0sec
450	27.6sec	27.0sec	26.3sec	27.7sec	33.4sec	26.0sec

Table 4.8 FVS-LDA computational load in terms of the mean iteration running time.

It is interesting that the GDA running times are not affected significantly for different values of eigenvectors considered unlike for the FVS-LDA scheme. That is expected since, at each iteration we deal with the decomposition of the kernel matrix based on the total amount of the available training data. Note that we do not consider code optimization or machines performance as a basis for comparing the classification schemes. However, we should expect to see the significantly reduced computational cost for FVS-LDA over GDA reflected in the amount of time it takes when both algorithms are executed on the same platform.

C. CONCLUSIONS

This chapter presented the classification performances obtained for the Bayes Classifier and the FVS-LDA schemes. In the former case, we considered the goodness-of-fit test for marginal densities and we discussed the peaking effect in the features selection. Next we introduced two different experiments for evaluating classification schemes, namely, an identification experiment and a verification experiment. The associated performance metrics Cumulative Match Score (CMS) and the Receiver Operating Characteristic (ROC) curves were defined and the concept of confidence intervals for a non-normal distributed population also discussed. We considered three different distance metrics in conjunction with two types of kernel functions, and applied kernel function parameter tuning following a five-fold cross-validation scheme.

In addition, we compared GDA and FVS-LDA classification performances. First, we applied the 5x2cv paired *t* test to investigate the impact of the selected number of features/eigenvectors on performance differences. Results showed that selecting between one third and a half of the feature vectors in the FVS-LDA scheme leads to statistically non-significant performance differences between FVS-LDA and GDA approaches when the same number of eigenvectors and well tuned kernel function parameters are used. Second, the verification experiment was applied to both schemes, and EER results show that the GDA scheme outperforms the FVS-LDA approach for that type of scenario. Finally, computational time issues were considered for both schemes and we noted a significantly reduced computational cost for the FVS-LDA implementation over that observed for the GDA scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

This study extended previous research conducted by [Domboulas, 2004] that investigated a nonlinear kernel-based classification scheme, the Generalized Discriminant Analysis (GDA) proposed by Baudat and Anouar [Baudat, 2000]. The GDA scheme was applied to an IR database, first generated by Pereira [Pereira, 2002] to include fourteen adult subjects collected in a controlled indoor environment and further expanded by Lee to a total of fifty adult subjects [Lee, 2004]. First, two widely applied linear classification schemes that had initially been applied in Pereira's and Lee's studies were reviewed, namely, the Principal Components Analysis (PCA) and the Linear Discriminant Analysis (LDA). Second, a linear classification scheme, the Bayes Classification algorithm, was considered in this study, to provide a baseline benchmarking performance. Third, the basic idea behind nonlinear kernel-based schemes was introduced via the GDA scheme review. Next, we considered an approach recently proposed by Baudat for nonlinear kernel methods which is based on a Feature Vector Selection (FVS) data selection process [Baudat, 2003]. Finally, the 5x2cv paired t-test, a statistical test based on estimated classification rates was applied to compare relative performances between GDA and FVS approaches to "determine whether one learning algorithm outperforms another on a particular learning task" [Dietterich, 1998].

We considered two different types of experiments to evaluate the performance of the FVS method followed by the classic linear classification LDA scheme. Specifically, identification and verification experiments along with their performance metrics, the Cumulative Match Characteristic (CMC) and the Receiver Operating Characteristic (ROC) curves, were investigated. Kernel-functions tuning issues were discussed, and we selected parameters for the Gaussian and the polynomial kernels in conjunction with three different types of distances (i.e., the Euclidean norm, the Euclidean angle, and the Mahalanobis angle) using a 5-fold cross-validation scheme. Results showed that the FVS scheme followed by the classic linear classification scheme LDA can achieve performances similar to those obtained with the GDA method at a significantly reduced computational cost. In fact, FVS-LDA turns out to be a good approximation of its

"kernelized" GDA version [Baudat, 2003]. Specifically selecting the number of features for the FVS-LDA scheme from a third to a half of the available features results in close to the standard GDA performance.

Our research concludes that, from a cost perspective, a low-resolution uncooled IR camera in conjunction with a low computational-cost classification scheme can be embedded in a robust face recognition system to efficiently address the issue of authentication in security-related tasks.

Future directions of this work may focus on some of the deficiencies of the kernel-based algorithms, namely the kernel functions parameters selection based on cross-validation schemes. This method is computationally expensive and does not guarantee the optimal selection of the parameters or optimal kernel mapping functions. Recently data-dependent kernel optimization models have been developed and proposed in the literature, [Xiong, 2005] and [Wang, 2005] to address these issues. The proposed models address those issues effectively by optimizing the measure of class separation in the feature space or generalizing classical kernel mapping functions to data-dependent kernels.

APPENDIX A. MATHEMATICAL BACKGROUND

This appendix presents the D'Agostino-Pearson K2 test which is a goodness-of-fit technique used to check whether class features are normal or not. That test was applied in the Bayesian classifier implementation to evaluate whether the features selected for representing the IR images could be considered as likely multivariate normal distributed random variables or not.

According to [Gnanadesikan, 1977], "although marginal normality does not imply joint normality, the presence of many types of non-normality is often reflected in the marginal distribution as well." Thus, in practice, evaluating the normality of multidimensional data should, in the first place, be evaluated for the marginal normality of the observations on each of the variables [Gnanadesikan, 1977].

To this end, a goodness-of-fit test for univariate normality may be applied first to assess the normality of the marginal distributions. We selected the D'Agostino-Pearson K2 test over the Kolmogorov-Smirnov and the Shapiro-Wilk normality tests because reports show that the latter tests do not work well (see [D'Agostino, 1986, Chapter 9] or [Graphpad, 2006] for further details).

A. ANALYSIS

1. D'Agostino-Pearson K2 Test

The D'Agostino-Pearson normality test is based on the standardized third and forth moments, better known as skewness and kurtosis, respectively, and defined as:

Skewness:
$$\sqrt{\beta_1} = \frac{E(X-\mu)^3}{\sigma^{3/2}}, \tag{A.1}$$

Kurtosis:
$$\beta_2 = \frac{E(X-\mu)^4}{\sigma^4}. \tag{A.2}$$

Note that $\sqrt{\beta_1} = 0$ and $\beta_2 = 3$ indicate a normal distribution. Pearson first used the concepts of skewness and kurtosis to evaluate deviation from normality. The formulas for computing those parameters from the sample data of size *n* are listed below [D'Agostino, 1986, Chapter 9]:

$$\sqrt{b_1} = m_3 / m_2^{\frac{3}{2}}, \tag{A.3}$$

$$b_2 = m_4 / m_2^{\,2}, \tag{A.4}$$

where

$$m_k = \sum (X - \bar{X})^k / n, k > 1 \tag{A.5}$$

and

$$\bar{X} = \sum X / n. \tag{A.6}$$

D'Agostino (1970) [D'Agostino, 1986, chap. 9] derived an approximately normal variable Z_1, based on the skewness parameter $\sqrt{b_1}$, with zero mean and unit variance as follows:

- First, compute $\sqrt{b_1}$ from the sample data using equations (1.4) to (1.7),
- Next, compute

$$Y = \sqrt{b_1}\left\{\frac{(n+1)(n+3)}{6(n-2)}\right\}^{\frac{1}{2}}, \tag{A.7}$$

$$\beta_2 = \frac{3(n^2+27n-70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)}, \tag{A.8}$$

$$W^2 = -1 + \left\{2(\beta_2 - 1)\right\}^{\frac{1}{2}}, \tag{A.9}$$

$$\delta = 1/\sqrt{\log W}, \tag{A.10}$$

$$\alpha = \left\{2/(W^2 - 1)\right\}^{\frac{1}{2}}, \tag{A.11}$$

- Last, compute the variable

$$Z_1 = \delta \log\left[Y/\alpha + \left\{(Y/\alpha)^2 + 1\right\}^{\frac{1}{2}}\right]. \tag{A.12}$$

Similarly, Anscombe and Glynn (1983) [D'Agostino, 1986, chap. 9] derived an approximately normal variable Z_2, based on the kurtosis quantity b_2, with zero mean and unit variance as follows:

- First, compute b_2 from the sample data using equations (1.4) to (1.7),
- Next, compute the mean and the variance of the parameter b_2, which leads to

$$E(b_2) = \frac{3(n-1)}{n+1}, \tag{A.13}$$

and

$$\text{var}(b_2) = \frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)}, \tag{A.14}$$

- Compute the standardized value of b_2,

$$x = \frac{b_2 - E(b_2)}{\sqrt{\text{var}(b_2)}}, \tag{A.15}$$

- Compute the third standardized moment of b_2,

$$\sqrt{\beta_1(b_2)} = \frac{6(n^2-5n+2)}{(n+7)(n+9)}\sqrt{\frac{6(n+3)(n+5)}{n(n-2)(n-3)}}, \tag{A.16}$$

- Compute

$$A = 6 + \frac{8}{\sqrt{\beta_1(b_2)}}\left[\frac{2}{\sqrt{\beta_1(b_2)}} + \sqrt{\left\{1 + \frac{4}{\beta_1(b_2)}\right\}}\right]. \tag{A.17}$$

- Last, compute

$$Z_2 = \left(\left(1 - \frac{2}{9A}\right) - \left[\frac{1-(2/A)}{1+x\sqrt{\{2/(A-4)\}}}\right]^{\frac{1}{3}}\right) / \sqrt{\{2/(9A)\}}\,. \tag{A.18}$$

D'Agostino and Pearson (1973) [D'Agostino, 1986, chap. 9] show that the parameter

$$K^2 = Z_1^2 + Z_2^2 \tag{A.19}$$

has a chi-square distribution with two degrees of freedom when the underlying data is normal. Thus, the D'Agostino and Pearson test evaluates how well the K^2 variable approximates the chi-square distribution for a given confidence level.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB SOURCE CODE

Matlab programming utilized in our study is provided in this appendix. Programs included were developed by the author or borrowed from other sources, reported in the references. Some programs were also modified accordingly to fulfill the requirements of our study. Functions or programs that have not been modified are only mentioned with no further details. Finally, a detailed modified version of the original software implementation is provided.

Below is listed the software implementation of the main experiments carried out in this study. First, the main body of the experiment is listed, second the associated programs/functions are also provided.

A. DESCRIPTION

1. Bayes Classifier Algorithm

a. Normality Test

- *IRdataNormalityTest.m*: Investigates whether training images features projected onto the PCA space are samples of a normal distribution before applying Bayes Classification scheme, utilizing the D'Agostino-Pearson's K2 test.
- *DagosPtest.m*: Applies the D'Agostino-Pearson's K2 test, [Trujillo-Ortiz and Hernandez-Walls, 2003].
- *myseparation.m*: Randomly separates data into training and testing.

b. Bayes-PCA Comparison

- *BayesVsPCA.m*: Compares the Bayes Classification scheme to the PCA approach with a fixed number of fifteen features (i.e. singular values) selected, in terms of the CMC plots (i.e. rank scores). The effect of varying the features in Bayes scheme is also illustrated.
- myseparation.m: As above.
- *RankScore.m*: Computes the cumulative matching scores (i.e. rank scores) for any testing image.
- *confint.m*: Returns the confidence intervals for the estimated classification rates computed over 1,000 iterations.

2. Kernel-Function Parameters Tuning Issues

a. FVS-LDA on Iris Data

- *IrisdataFVS-LDA.m*: Implements the FVS-LDA scheme on the Iris data for different values of feature vectors (FVs) and variance values of the Gaussian Kernel mapping. The effect of kernel parameters selection in class separation is also considered.
- *FVSselection2.m:* Computes the FVS for a set of M input vectors having N dimensions. The data must be a matrix MxN. A maximum number of selected FV can be defined, as well as a maximum for the global fitness function (up to 1). The function returns a data structure with the list of the FVs and the explicit projection of all the input data into the FV subspace of F, (G.Baudat and F.Anouar, pers. comm. 2004).
- *FVSProjection2:* Use the data structure from FVSelection2 to project any data (a matrix) into the FV subspace of F, (G.Baudat and F.Anouar, pers. comm. 2004).
- *FVSKernel2.m:* This function evaluates the dot products into F using the selected kernel or a set of vectors. It is used by the previous functions, (G.Baudat and F.Anouar, pers. comm. 2004).
- *dist.m:* Compute Euclidean distances for some kernels, (G.Baudat and F.Anouar, pers. comm. 2004).
- *invrec.m:* Recursive inversion of a kernel matrix, (G.Baudat and F.Anouar, pers. comm. 2004).
- *fld.m:* Applies the LDA method on the data, [Lee, 2004].
- *sortem.m:* Sorts eigenvectors by decreasing eigenvalue magnitude [Lee, 2004].

b. Cross-Validation Scheme

- *crossvalidate.m:* Performs the K-fold hold-out cross-validation scheme to obtain the global parameters associated with the kernel mapping functions.
- *discriminate.m:* This function computes the classification rates for a pair of training and testing sets for a given distance metric.
- FVSselection2.m: As above.
- FVSProjection2: As above.
- FVSKernel2.m: As above.
- *dist2.m:* As above.
- *invrec.m:* As above.
- *fld.m:* As above.
- *sortem.m:* As above.

- myseparation.m: As above.
- RankScore.m: As above.

3. FVS-LDA Algorithm Implementation on IR Data, the Identification Type of Experiment

- *FVS_LDA_IRdata_Identification.m:* Implements the FVS-LDA scheme on IR data using the global parameters obtained via cross-validation. The identification type of experiment is considered for performance evaluation. The associated confidence intervals for a 95% significance level are estimated.
- FVSselection2.m: As above
- FVSProjection2: As above
- FVSKernel2.m: As above
- dist2.m: As above
- invrec.m: As above
- confint.m: As above
- discriminate.m: As above
- *fld.m:* As above
- sortem.m: As above
- myseparation.m: As above
- RankScore.m: As above

4. FVS-LDA and GDA Algorithms Implementation on IR Data, the Verification Type of Experiment

a. FVS-LDA Scheme

- *FVS_LDA_IRdata_Verification.m:* Implements the FVS-LDA scheme on IR data using the global parameters obtained via cross-validation. The verification type of experiment is considered for performance evaluation.
- FVSselection2.m: As above
- FVSProjection2: As above
- FVSKernel2.m: As above
- dist2.m: As above
- invrec.m: As above
- confint.m: As above
- discriminate.m: As above
- *fld.m:* As above

- sortem.m: As above

- myseparation.m: As above
- RankScore.m: As above

b. GDA Scheme

- *GDA_IRdata_Verification.m:* Performs the GDA method, for various kernel functions and distances on IR data. The verification type of experiment is considered for performance evaluation.
- myseparation.m: As above
- *dataST.m*: Normalizes input data by setting mean to 0 and standard deviation to 1. (From [Baudat, October 2000].).
- *buildGDA_Opt.m*: Applies the GDA algorithm to the input data and creates the GDA data, which will be used for the input data projection (from [Domboulas, 2004]).
- *eigensystem.m*: Performs eigenvalue–eigenvector decomposition of the input matrix and sorts the eigenvectors by decreasing eigenvalue magnitude (From [Baudat, 2000].).
- *spreadGDA_Opt.m*: Projects the input data onto the GDA eigenvectors (from [Domboulas, 2004]).
- *KernelFunctiont.m*: Function called only when the Gaussian kernel function is selected by setting the parameter "degree" equal to 0 (From [Baudat, 2000].).

5. Comparison of the FVS-LDA and GDA Algorithms on Iris Data Based on Experimental Results of the Identification Experiment Results, the 5x2cv Paired *t*-Statistical Test

- *the5x2cvpaired_ttest.m:* Implements the 5x2cv paired t test to compare FVS-LDA versus GDA algorithm based on the estimated classification rates for various number of feature vectors / eigenvectors selected respectively.
- myseparation.m: As above
- buildGDA_Opt.m: As above
- spreadGDA_Opt.m: As above
- Datast.m: As above
- eigensystem.m: As above
- KernelFunction.m: As above
- dist2.m: As above
- *fld.m:* As above

- FVSselection2.m: As above
- FVSKernel2.m: As above
- FVSProjection2: As above
- invrec.m: As above
- sortem.m: As above
- myseparation.m: As above.

B. MATLAB CODE LISTING

- IRdataNormalityTest.m:

```
%                                                                       %
************************************************************************
% % File name:         IRdataNormalityTest.m
% % Thesis Advisor: Prof.M.P.Fargues Naval Postgraduate School
% % Author:            Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
% % Date:              November 2005
% % Description:       Implements D'Agostino-Pearson's K2 test to check
% %                    whether the features of the IR training data projected
% %                    onto the subspace obtained when the PCA step is applied
% %                    before applying Bayes Classifier method, are normally
% %                    distributed.
% %                    A 95% significance for the test level is considered.
% % Parameters        Number of singular values (max 17 for a 60-40%separation
% %                    into training and testing data respectively).
% % Outputs           1.Plots the number of populations (i.e. features) per
% %                    class that FAIL the test.
% % Functions used:  DagosPtest.m myseparation.m
%
%************************************************************************
**

clc;
clear;
load A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A_all);
clear  A_all;

%CONSTANTS DECLARATION

Train_Class_Size= 18;       % Number of Training Images per Class
Test_Class_Size = 12;       % Number of Testing Images per Class
Num_Train_Imag = 900;       % Total Number of Training Images
Num_Test_Imag = 600;        % Total Number of Testing Images
```

```
Num_Class = 50;            % Total Number of Classes
Sample_Class=30;           % Samples per class
train=[];
test=[];
[row,col] = size(Atemp);


% % DATA SEPARATION INTO TRAINING & TESTING
[A1,A2]=myseparation(Atemp,Sample_Class,Num_Class,Train_Class_Size);

%APPLY THE PCA USING THE SNAPSHOT METHOD

X=A1-mean(mean(A1));
X=double(X);
[U,S,V] = svd(X,0); % produces the “economy size” decomposition. If X is m-by-n with
m > n, then only the
                % first n columns of U are computed and S is n-by-n.

% figure
% plot(diag(S))
% grid
% title('Singular Values of Training Data SVD')

%SELECT THE NUMBER OF SINGULAR VALUES
for c=1:50                      %# classes
   c
singular_values=12;
U_proj=U(:,(1:singular_values));    % projection matrix
X_proj=U_proj'*X;                   % projected data
Class_c_proj=X_proj(:,(18*(c-1)+1:18*c));

%GOODNESS OF FIT TEST

%APPLY THE D'Agostino-Pearson's K2 test for assessing normality
%of the Class_c_proj data where each row is a feature
%using skewness and kurtosis.
   for     k=1:singular_values; % feature vector coordinates
           x_1=sort(Class_c_proj(k,:)');
           H(k,c)=DagosPtest(x_1,0.05);
           % Description performs the D'Agostino-Pearson's K2 test on the input data vector
X
           % and returns H, the result of the hypothesis test.
           % The result H is 1 if you can reject the hypothesis that X has a normal
distribution,
           % or 0 if you cannot reject that hypothesis.
           % you reject the hypothesis if the test is significant at the 5% level.
```

```
   end
end
figure
bar(sum(H))
axis([0,50,0,12])
set(gca,'xtick',[0:5:50])
set(gca,'ytick',[0:2:12])
set(gca,'fontsize',14)
xlabel('class index')
ylabel('# of populations fail the test')
title('D Agostino-Pearsons K2 goodness of fit test')
h1=legend('fail');
set(h1,'fontsize',12)
grid
```

- DagosPtest.m:

```
function [x] = DagosPtest(X,alpha,Fail)
% D'Agostino-Pearson's K2 test for assessing normality of data using skewness and kurtosis.
%
%   Syntax: function [DagosPtest] = DagosPtest(X,alpha)
%
%    Inputs:
%        X - data vector.
%      alpha - significance level (default = 0.05).
%
%    Outputs:x %%Modified by Alexandropoulos
%        - Whether or not the normality is met.
%        - x=1 normality NOT met
%        - x=0 normality met
%
%
%   Example: From the example 6.8 of Zar (1999, p.89), we are interested to test
%           whether or not the data are normally distributed using the D'Agostino-
%           Pearson test with a significance level = 0.05.
%
%                  x      Frequency
%               ----------------------
%                  63         2
%                  64         2
%                  65         3
%                  66         5
%                  67         4
%                  68         6
%                  69         5
%                  70         8
```

```
%                 71         7
%                 72         7
%                 73        10
%                 74         6
%                 75         3
%                 76         2
%               ----------------------
%
%         Data matrix must be:
%
X=[63;63;64;64;65;65;65;66;66;66;66;66;67;67;67;67;68;68;68;68;68;68;69;69;69;69;69
;
%
70;70;70;70;70;70;70;70;71;71;71;71;71;71;71;72;72;72;72;72;72;72;73;73;73;73;73;73;
%     73;73;73;73;74;74;74;74;74;74;75;75;75;76;76];
%
%    Calling on Matlab the function:
%           DagosPtest(X)
%
%      Answer is:
%
% D'Agostino-Pearson's test to assessing normality: X2= 3.1397, df= 2
% Probability associated to the Chi-squared statistic = 0.2081
% The sampled population is normally distributed.
%

%  Created by A. Trujillo-Ortiz and R. Hernandez-Walls
%           Facultad de Ciencias Marinas
%           Universidad Autonoma de Baja California
%           Apdo. Postal 453
%           Ensenada, Baja California
%           Mexico.
%           atrujo@uabc.mx
%
%  September 11, 2003.
%
%  To cite this file, this would be an appropriate format:
%  Trujillo-Ortiz, A. and R. Hernandez-Walls. (2003). DagosPtest: D'Agostino-Pearson's
K2 test for
%    assessing normality of data using skewness and kurtosis. A MATLAB file. [WWW
document]. URL
%
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3954&objec
tType=FILE
%
%  References:
```

```
%
%  Zar, J. H. (1999), Biostatistical Analysis (2nd ed.).
%          NJ: Prentice-Hall, Englewood Cliffs. p. 89.
%

if nargin < 2,
  alpha = 0.05;
end;

if (alpha <= 0 | alpha >= 1)
  fprintf('Warning: significance level must be between 0 and 1\n');
  return;
end;

n = length(X);

[c,v]=hist(X,X);  %record of data in a frequency table form
nc=find(c~=0);
c=[v(nc) c(nc)'];

x = c(:,1);
f = c(:,2);
s1 = f'*x;
s2 = f'*x.^2;
s3 = f'*x.^3;
s4 = f'*x.^4;
SS = s2-(s1^2/n);
v = SS/(n-1);
k3 = ((n*s3)-(3*s1*s2)+((2*(s1^3))/n))/((n-1)*(n-2));
g1 = k3/sqrt(v^3);
k4   =   ((n+1)*((n*s4)-(4*s1*s3)+(6*(s1^2)*(s2/n))-((3*(s1^4))/(n^2)))/((n-1)*(n-2)*(n-
3)))-((3*(SS^2))/((n-2)*(n-3)));
g2 = k4/v^2;
eg1 = ((n-2)*g1)/sqrt(n*(n-1));  %measure of skewness
eg2 = ((n-2)*(n-3)*g2)/((n+1)*(n-1))+((3*(n-1))/(n+1));  %measure of kurtosis

A = eg1*sqrt(((n+1)*(n+3))/(6*(n-2)));
B = (3*((n^2)+(27*n)-70)*((n+1)*(n+3)))/((n-2)*(n+5)*(n+7)*(n+9));
C = sqrt(2*(B-1))-1;
D = sqrt(C);
E = 1/sqrt(log(D));
F = A/sqrt(2/(C-1));
Zg1 = E*log(F+sqrt(F^2+1));

G = (24*n*(n-2)*(n-3))/((n+1)^2*(n+3)*(n+5));
H = ((n-2)*(n-3)*abs(g2))/((n+1)*(n-1)*sqrt(G));
```

```
J = ((6*(n^2-(5*n)+2))/((n+7)*(n+9)))*sqrt((6*(n+3)*(n+5))/((n*(n-2)*(n-3))));
K = 6+((8/J)*((2/J)+sqrt(1+(4/J^2))));
L = (1-(2/K))/(1+H*sqrt(2/(K-4)));
Zg2 = (1-(2/(9*K))-L^(1/3))/sqrt(2/(9*K));

K2 = Zg1^2 + Zg2^2;  %D'Agostino-Pearson statistic
X2 = K2;  %approximation to chi-distribution
df = 2;  %degrees of freedom

P = 1-chi2cdf(X2,df);  %probability associated to the chi-squared statistic
if P >= alpha;
 x=0;%%Modified by Alexandropoulos
%disp('P'); %disp('The sampled population is normally distributed.');
else
% fprintf('D"Agostino-Pearson"s test to assessing normality: X2= %3.4f, df=%2i\n',
X2,df);
% fprintf('Probability associated to the chi-squared statistic = %3.4f\n', P);
%fprintf('With a given significance = %3.3f\n', alpha);
% disp('FAIL') %disp('The sampled population is NOT!!!!!!! normally distributed.');
x=1;%%Modified by Alexandropoulos
end;
```

- myseparation.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[training,testing]=myseparation(X,Sample_Class,Num_Class,Train_Class_Size)
%       This function separates the initial data in matrix X into training and
%       testing data
%       Date:April 2006
%
%       inputs:         X,initial data
%                       Sample_Class,samples collected per class
%                       Num_Class,number of classes
%                       Train_Class_Size,number of samples per class for training
%
%       outputs:        training,the training data
%                       testing,the teasting data
%
% Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[f,t]=size(X);

A_1=reshape(X,f,Sample_Class,Num_Class);
[row,col] = size(A_1);

pose=3;
```

```
A_neutral=A_1(:,1:10,:);
A_u=A_1(:,11:20,:);
A_smile=A_1(:,21:30,:);

p = randperm(Sample_Class/pose);
ptrain=p(1:Train_Class_Size/pose);
ptest=p((Train_Class_Size/pose)+1:Sample_Class/pose);

Test_Class_Size=Sample_Class-Train_Class_Size;

for k=1:Num_Class;

   for i=1:Train_Class_Size/pose;
   train_neutral(:,i,k)=A_neutral(:,ptrain(i),k);
   train_u(:,i,k)=A_u(:,ptrain(i),k);
   train_smile(:,i,k)=A_smile(:,ptrain(i),k);
   end

   for ii=1:Test_Class_Size/pose;
   test_neutral(:,ii,k)=A_neutral(:,ptest(ii),k);
   test_u(:,ii,k)=A_u(:,ptest(ii),k);
   test_smile(:,ii,k)=A_smile(:,ptest(ii),k);
   end

end
train=[train_neutral train_u train_smile];
test=[test_neutral test_u test_smile];
Train_data=reshape(train,row,1,Train_Class_Size*Num_Class);
Test_data=reshape(test,row,1,Test_Class_Size*Num_Class);

training=squeeze(Train_data);        %training data each column is a sample
testing=squeeze(Test_data);          %testing data each column is a sample
```

- BayesVsPCA.m:

```
%                                                                              %
********************************************************************************
% % File name:        BayesVsPCA
% % Thesis Advisor:  Prof.M.P.Fargues Naval Postgraduate School
% % Author:          Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
% % Date:            November 2005
% % Description:     Implements the Bayes Classification scheme and compares
% %                  the results for those obtained with the PCA scheme when
% %                  15 singular values are selected. Performances compared
% %                  using the Rank one score. The associated confidence
```

```
% %                    intervals for a 95% significance level are estimated.
% % Parameters        1.Number of iterations
% % Outputs           1.CMC plots
% %                   2.Per class classification performance plots
% %                   3.Rank one performance metrics
% %                   4.Confidence Intervals
% % Functions used:  RankScore.m, confint.m, myseparation.m
%
%**************************************************************************
**


clc;
clear;
load A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A);
clear  A_all;

%CONSTANTS DECLARATION

Train_Class_Size= 18;                % Number of Training Images per Class
Test_Class_Size = 12;                % Number of Testing Images per Class
Num_Class = 50;                      % Total Number of Classes
Sample_Class=30;                     % Samples per class
Num_Train_Imag =Train_Class_Size*Num_Class; % Total Number of Training Images
Num_Test_Imag =Test_Class_Size*Num_Class;   % Total Number of Testing Images
Num_iter=2;                          %Number of Iterations 1000 used
train=[];test=[];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for iterations=1:Num_iter          %ITERATIONS LOOP(OUTER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % DATA SEPARATION INTO TRAINING & TESTING
[A1,A2]=myseparation(Atemp,Sample_Class,Num_Class,Train_Class_Size);

% % APPLY THE PCA USING THE SNAPSHOT METHOD

X=A1-mean(mean(A1));
X=double(X);
[U,S,V] = svd(X,0); % produces the “economy size” decomposition. If X is m-by-n with
m > n, then only the
```

```
                % first n columns of U are computed and S is n-by-n.
Test_data=A2;Train_data=A1;

clear A A1 A2  S  V  dim1 dim2 test train

% % SELECT THE NUMBER OF SINGULAR VALUES

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for singular_values=1:15;                % SINGULAR VALUES LOOP INNER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
singular_values
U_proj=U(:,(1:singular_values));          % projection matrix
X_proj=U_proj'*X;                        % projected data
X_train_proj=U_proj'*Train_data;              % Projected Training Data 18 samples per
class
X_test_proj=U_proj'*Test_data;            % Projected Testing Data 12 samples per class
                                % In total 30 samples per class

% % DISCRIMINANT FUNCTION g

for i=1:Num_Class;
   cl=X_train_proj(:,(1+Train_Class_Size*(i-1):Train_Class_Size*i));
   mu(:,i)=mean(cl');            % mean vector of each class
   Sig(:,:,i)=cov(cl');          % covariance matrix of class
   Sigma=cov(cl');               % covariance matrix of class
   D_Sigma(i)=det(Sigma);        % its determinant
   Inv_S(:,:,i)=inv(Sigma);
   c(i)=-0.5*log(det(Sigma)+eps);  % constant of each class
end

[row,col]=size(X_test_proj);

for n=1:col;
   for i=1:Num_Class;
      %% Bayes
      g(i,n,singular_values)=-0.5*(X_test_proj(:,n)-
mu(:,i))'*Inv_S(:,:,i)*(X_test_proj(:,n)-mu(:,i))+c(i);
      %% PCA
      g_PCA15(i,n,singular_values)=-0.5*(X_test_proj(:,n)-mu(:,i))'*(X_test_proj(:,n)-
mu(:,i));
   end
end

for n=1:col;
```

```
[value(n),index(n)]=max(g(:,n,singular_values));
[valuePCA15(n),indexPCA15(n)]=max(g_PCA15(:,n,singular_values));
end

clear mu Sig Inv_S  X_train_proj cl Sigma D_Sigma c row col value


%plot(index)

% % CONFUSION MATRIX

for i=1:Num_Class
   a(:,i)=index(1+Test_Class_Size*(i-1):Test_Class_Size*i);
   aPCA15(:,i)=indexPCA15(1+Test_Class_Size*(i-1):Test_Class_Size*i);
   for ii=1:Num_Class
   I(ii,i,iterations)=sum(a(:,i)==ii);                 %Bayes
   IPCA15(ii,i,iterations)=sum(aPCA15(:,i)==ii);       %PCA
   end

per_class_classification_rate(i,singular_values,iterations)=I(i,i,iterations)/Test_Class_Siz
e;

per_class_classification_ratePCA15(i,singular_values,iterations)=IPCA15(i,i,iterations)/
Test_Class_Size;
end
overall_classification_rate(iterations,:,singular_values)=100*sum(diag(I(:,:,iterations)))/
Num_Test_Imag;
overall_classification_ratePCA15(:,singular_values)=100*sum(diag(IPCA15(:,:,iterations
)))/Num_Test_Imag;
clear  a

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%REFERS TO THE SINGULAR VALUES (END OF INNER LOOP)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

performance(iterations)=max(overall_classification_rate(iterations,:,:));
performancePCA15(iterations)=max(overall_classification_ratePCA15);

%Best performance per each iteration i.e optimum singular value to be used
[best_rate,optimum_singular_value]=max(overall_classification_rate(iterations,:,:));
[best_rate,optimum_singular_valuePCA15]=max(overall_classification_ratePCA15);
```

```
Performance_per_class=squeeze(per_class_classification_rate(:,optimum_singular_value,
iterations));
Performance_per_classPCA15=squeeze(per_class_classification_ratePCA15(:,optimum_
singular_valuePCA15,iterations));

max_index(iterations)=optimum_singular_value;          %indeces where max occurs
max_indexPCA15(iterations)=optimum_singular_valuePCA15;  %indeces where max
occurs


%Associated disriminant function for each performance case
g_11=g(:,:,11)';                                %Bayes_11
g_15=g(:,:,15)';                                %Bayes_15
g_opt=g(:,:,optimum_singular_value)';                   %Bayes_optimum
g_optPCA15=g_PCA15(:,:,optimum_singular_valuePCA15)';      %PCA_15

clear optimum_singular_value per_class_classification_rate value
clear X_proj U_proj best_rate

% % %RANK SCORE(CUMULATIVE MATCHING SCORE)

[Rank]=RankScore(g_11,Num_Class,Test_Class_Size,Num_Test_Imag);
Rank_Bayes_11(iterations,:) = Rank;

[Rank]=RankScore(g_15,Num_Class,Test_Class_Size,Num_Test_Imag);
Rank_Bayes_15(iterations,:) = Rank;

[Rank]=RankScore(g_opt,Num_Class,Test_Class_Size,Num_Test_Imag);
Rank_Bayes(iterations,:) = Rank;

[Rank]=RankScore(g_optPCA15,Num_Class,Test_Class_Size,Num_Test_Imag);
Rank_PCA15(iterations,:) = Rank;

clear temp1 temp2 I_rank Rank Prob T_B ii g_opt p
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%REFERS TO THE NUMBER OF ITERATIONS (END OF OUTER LOOP)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % CONFIDENCE INTERVALS

%Bayes 11 singular values
performance_Bayes_11=overall_classification_rate(:,:,11);
interval_11=confint(performance_Bayes_11,0.05,iterations);
```

```
%Bayes 15 singular values
performance_Bayes_15=overall_classification_rate(:,:,15);
interval_15=confint(performance_Bayes_15,0.05,iterations);

%Bayes optimum singular values
performance_Bayes_opt=performance;
interval_opt=confint(performance_Bayes_opt,0.05,iterations);



% save Bayes_Classifier


figure(1)
stem(mean(Rank_Bayes_15));grid
title(' Bayes Classifier based on the 15 singular values of the per class Covariance matrix SVD')
ylabel('Cummulative Rank Score')
xlabel('Rank Index')
axis ([-2 52,0 1.2])
text(0,1.05,'Number of Singular values fixed to 15')

figure(2)
stem(mean(Rank_Bayes));grid
title(' Bayes Classifier based on the OPTIMUM singular values of the per class Covariance matrix SVD')
ylabel('Cummulative Rank Score')
xlabel('Rank Index')
axis ([-2 52,0 1.2])
text(0,1.175,'Number of Singular values vary,')
text(0,1.1,'those that obtain the max Classification rate selected')

figure(3)
stem(Performance_per_class');grid
overall_classification_performance=mean(performance)
title(' Bayes Classifier//Classification per Class Performance')
ylabel('Classification Performance')
xlabel('Class Index')
axis ([-2 52,0 1.2])


figure(4)
stem(mean(Rank_PCA15));grid
title(' PCA Classifier based on the top 15 eigenvalues training data SVD')
```

```
ylabel('Cummulative Rank Score')
xlabel('Rank Index')
axis ([-2 52,0 1.2])
text(0,1.05,'Number of Singular values fixed to 15')

figure(5)
stem(Performance_per_classPCA15');grid
overall_classification_performancePCA15=mean(performancePCA15)
title(' PCA15 Classifier//Classification per Class Performance')
ylabel('Classification Performance')
xlabel('Class Index')
axis ([-2 52,0 1.2])

figure(6)
stem(mean(Rank_Bayes_11));grid
title(' Bayes Classifier based on the 11 singular values of the per class Covariance matrix
SVD')
ylabel('Cummulative Rank Score')
xlabel('Rank Index')
axis ([-2 52,0 1.2])
text(0,1.05,'Number of Singular values fixed to 11')

figure(7)
stem(squeeze(mean(overall_classification_rate(:,:,:),1)))
axis([0,20,0,100])
title('Bayes Classifier,the “peaking effect”.')
xlabel('features selected(i.e. singular values of the class covariance matrix)')
ylabel('Classification rate')
grid
```

- RankScore.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Rank] = RankScore(g,Num_Class,Test_Class_Size,Num_Test_Imag)
%		g is the discrimination function
%		Rank is the Cumulative Rank Score
%		Created by Domboulas [Domboulas, 2004]
%		Modified by Alexandropoulos
%%% Assign 1-50 Class Indices for each of the test Images
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% T_B = [1,1,1,1,1,1,1,1,1,1,1,1];
```

```
T_B=ones(1,Test_Class_Size);%%%%Modified by Alexandropoulos to include
% any per centage separation
for m = 1 : Num_Class;
T_B( ( Test_Class_Size * (m-1) + 1 ) : ( Test_Class_Size * m )   ) = m * T_B( 1:Test_Class_Size) ;
end; %%% T_B contains the class number of each testing image

%%% Sort Distances per Testing  Image - RANK
for k = 1 : Num_Test_Imag ;
temp1 = sort( g(k,:));
I_rank(k) = find(  (temp1 == g(k,T_B(k))*ones(1,Num_Class)) );
end;

%%% Find the probability of each of the 50 Ranks assigned to all the testing images
for k =1 : Num_Class ;
   temp2 = find( I_rank == k * ones( 1,Num_Test_Imag ) );
   Prob(k) = size( temp2 , 2 ) / Num_Test_Imag;
end;

%%%%  RANK SCORE Evaluation - Cummulative Probability of Ranks - Incresing Order
Rank(1) = Prob(1);
for k = 2 : Num_Class ;
   Rank( k ) = Rank( k-1 ) + Prob( k );  %% Rank is 1X50 with accumulated Probability
end;
```

- confint.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function interval=confint(overall_classification_rate,confidence,iterations)
%%% This function returns the confidence interval bounds using the
%%% percentile method for non normal distributed populations
%       inputs:         overall_classification_rate are the classification rates
%                       per each iteration
%                       confidence is the selected confidence level
%                       iterations refers to the total repetition of the
%                       classification experiment
%
%       outputs:        confidence interval bounds
%
% Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a1=sort(overall_classification_rate);
```

```
a2=floor((confidence)*0.5*iterations);
low=a1(:,a2);
high=a1(:,iterations-a2);
interval=[low high];
```

- IrisdataFVS-LDA.m

```
%                                                                                %
**********************************************************************
% % File name:       IrisdataFVS-LDA.m
% % Thesis Advisor: Prof.M.P.Fargues Naval Postgraduate School
% % Author:         Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
% % Date:           February 2006
% % Description:    Implements the FVS-LDA scheme on the Iris data for
% %                 different values of feature vectors FVs and the variance
% %                 of the Gaussian Kernel mapping. The effect of kernel
% %                 parameters selection in class separation is considered.
% %
% % Parameters      1.variance p1= sigma^2 for a Gaussian kernel mapping
% %                 2.Number of feature vectors FVs (max 150 i.e. the total
% %                   available training data samples)
% %                 3.global fitness (between 0 to 1)
% % Outputs         1.Scatter plots of the projected Iris data classes
% %
% % Functions used:  FVSelection2.m, testKFA2.m, FVSProjection2.m, fld.m
% %                 FVSKernel2.m, applyKFA2.m, dist.m, invrec.m, sortem.m
%
%**********************************************************************
**

clc;
clear;
load fisheriris;    %% Load the IrisData
% set up the global parameter p1= sigma^2 for a gaussian kernel
global p1;
p1=700;       % sigma^2 = p1
IrisData=meas;
dataL1=meas(1:50,:);    %setosa
dataL2=meas(51:100,:);  %versicolor
dataL3=meas(101:150,:); %virginica
dataL=[dataL1;dataL2;dataL3];

% launch the feature vector selection
```

```
fprintf('Feature Vector Selection in progress ...\n');
dataFVS=FVSelection2(meas,'gauss',15,1-eps);



 P=dataFVS.P;
% dataP1=P(1:50,:);    %setosa
% dataP2=P(51:100,:);  %versicolor
% dataP3=P(101:150,:); %virginica
%
c=ones(1,50);
C=[c 2*c 3*c];             %classes associated to numbers

[W,D]=fld(P',C);

X_proj=W'*P';           %data projected to fisher eigenspace
X_proj_C1=X_proj(:,(1:50));        %class 1
X_proj_C2=X_proj(:,(51:100));      %class 2
X_proj_C3=X_proj(:,(101:150));     %class 3

figure
scatter(X_proj_C1(1,:),X_proj_C1(2,:),'o')
hold on
scatter(X_proj_C2(1,:),X_proj_C2(2,:),'+','r')
hold on
scatter(X_proj_C3(1,:),X_proj_C3(2,:),'d')
hold off
grid
title('2-D Projected Iris training data using FVS-LDA,Gaussian Kernel')
legend('class 1','class 2','class 3',0)
```

- crossvalidate.m:

```
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% File name:        crossvalidate.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:          LTJG(E) Ioannis M.Alexandropoulos
%%                  Hellenic Navy
%% Date:             April 2006
%% Description:      Performs the hold out cross-validation scheme to
%%                  obtain the global parameters associated with the kernel
%%                  mapping functions
%% Parameters:       1. Kernel mapping function
%%                  2. Range of the kernel global parameters
%%                  3. Number of feature vectors selected
```

```
%%              4. Distance type to be used
%%              5. Number of segments used for the hold out scheme
%% Outputs:        1. Rank-(1-50) Scores
%%              2. Classification Performance per Class
%%              3.Classification error rates plots versus global
%%                parameters (i.e. order/variance)
%% Functions used:   FVSelection2.m, FVSKernel2.m,  fld.m,
%%              FVSProjection2.m, dist2.m,  sortem.m
%%              myseparation.m, discriminate.m , invrec.m
%%              RankScore.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc;
clear;
load A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A);
clear  A A_all;

% %CONSTANTS DECLARATION

Train_Class_Size= 24;                  % Number of Training Images per Class
Test_Class_Size = 6;                   % Number of Testing Images per Class
Num_Class = 50;                        % Total Number of Classes
Sample_Class=30;                       % Samples per class
Num_Train_Imag =Train_Class_Size*Num_Class; % Total Number of Training Images
Num_Test_Imag =Test_Class_Size*Num_Class;   % Total Number of Testing Images
Atemp=Atemp-mean(mean(Atemp));             % centering step i.e zero mean
train=[];test=[];

% % CROSS-VALIDATION

%Partitioning of the training data into S segments for use in cross-validation.
hold_out_ratio=Train_Class_Size/Test_Class_Size;
   var1=6:.4:10;
   var2=14:4:50;
   var=[var1 var2]
   for kk=1:length(var);
   kk

global p1;                 % sigma^2 = p1
p1=var(kk)*10^(6);           %varying the p1 parameter
segments=hold_out_ratio+1;     %S segments
```

```
for k=1:segments;
   k
[AA1,AA2]=myseparation(Atemp,Sample_Class,Num_Class,Train_Class_Size);% Selection of training and testing data

% %FEATURE VECTOR SELECTION
Features=350;
Global_Fitness=1-eps;
fprintf('Feature Vector Selection in progress ...\n'); % set up the global parameter p1= sigma^2 for a gaussian kernel
dataFVS=FVSelection2(AA1','gauss',Features,Global_Fitness);% Select feature vectors among the training data
S=dataFVS.S;                                % S is the new basis subspace spanned by the selected feature vectors
Ptrain=dataFVS.P;                           % Training Data projected onto the subspace of F
Ptest=FVSKernel2('gauss',AA2',AA1(:,S)')';          % Projection of all the testing data onto the subspace of F
                                            % the FVs space(explicit projection using the kernel
%After FVs selection using the non-linear kernel mapping data are now linear separable

% %APPLY FLD

%Create C matrix
for i=1:Num_Class;
   c(:,i)=i*ones(1,Train_Class_Size);
end
C=reshape(c,1,Train_Class_Size*Num_Class);
[W,D]=fld(Ptrain',C);
X_train_proj=W'*Ptrain';          % Projected Training Data onto FVs-LDA space
X_test_proj=W'*Ptest';            % Projected Testing Data onto FVs-LDA space
%RECALL that the dimension of the FVS-LDA subspace is always N-1 where N is
%the number of classes in our case N=50

% %DISCRIMINANT FUNCTION
Distance='euclidean';
[per_class_classification_rate,I,g]=discriminate(X_train_proj,X_test_proj,Distance);
%discriminate returns the per class classification rate,
%the confusion matrix I
%the discriminant's values g
overall_classification_rate(:,k)=100*sum(diag(I(:,:)))/Num_Test_Imag;

end
m_error_rate(:,kk)=100-mean(overall_classification_rate) %mean error rate as a function of p1 variation
clear p1                                    %pi is global so needs to be cleared
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%END OF CROSS VALIDATION SCHEME%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for kk=1:length(var)
   variance(:,kk)=var(kk)*10^(6);
end
upper=max(variance);down=min(variance);
range=[down upper];
figure(1)
   plot(variance,m_error_rate)
   title('Estimating the variance using the hold out method')
   xlabel('variance(p1)')
   ylabel('mean error rate')
   grid
[value index]=min(m_error_rate)
sigma_squared=variance(:,index)

% % Results returned in a structure

% % Results=struct('Segments',segments,'FVs',Features,'BestVariance',sigma_squared,
'Performance', value,'Var_Range',range,'Distance',Distance );
% %
% %
% % clear AA1 col AA2 dataFVS Atemp dim1 C dim2 D g I i Num_Class ii
Num_Test_Imag index Num_Train_Imag k Num_iter kk
% % clear Ptest Ptrain mu  S n Sample_Class overall_classification_rate Test_Class_Size
per_class_classification_rate
% % clear Train_Class_Size row W segments X_test_proj test X_train_proj train a value
c  cl
% %
% % save Trial_1
```

- discriminate.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function
[per_class_classification_rate,I,g]=discriminate(X_train_proj,X_test_proj,distance)
% This function computes the classification rate for the below selection of
% discrimination criteria
% Date:12 March 2006
%
% input: Xtrain- matrix containing the data projected in  linear separable subspace
%        matrix should be in the form samples-by-features
```

```
%       Xtest- matrix containing the data projected in  linear separable subspace
%       matrix should be in the form samples-by-features
%       discrimination-type of discriminatio function to be used
%
% output: Ptrain-Performance when training data are presented
%        Ptest-Performance when training data are presented
%
% Ioannis Alexandropoulos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %DISCRIMINANT FUNCTION
Train_Class_Size= 24;                   % Number of Training Images per Class
Test_Class_Size = 6;                    % Number of Testing Images per Class
Num_Class = 50;                         % Total Number of Classes

for i=1:Num_Class;
   cl=X_train_proj(:,(1+Train_Class_Size*(i-1):Train_Class_Size*i)); % class seperation
   mu(:,i)=mean(cl');                                  % mean vector of each class or Class
Centroids
end

%used for mahalanobis angular
matrix_1=cov(X_train_proj');
sigma=sqrt(diag(matrix_1));
matrix_2=diag((sigma).^-1);

[row,col]=size(X_test_proj);


for n=1:col;
   for i=1:Num_Class;
       switch lower(distance)   % selection of the discrimination function

       case 'euclidean'     % euclidean (norm_2) discriminant function
       g(i,n)=-norm(X_test_proj(:,n)-mu(:,i),2);

       case 'mah_angular'   % mahalanobis angular
       g(i,n)= -acos( ( ( X_test_proj(:,n)' * matrix_2 ) * ( matrix_2 * mu(:,i) ) ) / ( norm(
X_test_proj(:,n).*((sigma).^-1)    ,2    )    *    norm(    mu(:,i).*((sigma).^-1)    ,2    )    )    );
% MAHALANOBIS ANGULAR DISTANCE

       case 'euc_angular'   % euclidean angular
       g(i,n)= -acos( ( ( X_test_proj(:,n)' *  mu(:,i) ) ) / ( norm( X_test_proj(:,n),2 ) * norm(
mu(:,i),2 ) ) ); % euclidean angular
```

```
      otherwise
      disp('Discriminat function type unknown')
      end
   end
end

for n=1:col;
[value(n),index(n)]=max(g(:,n)); % find where max occurs
end
% plot(index);

% %CONFUSION MATRIX

for i=1:Num_Class
   a(:,i)=index(1+Test_Class_Size*(i-1):Test_Class_Size*i);
   for ii=1:Num_Class
   I(ii,i)=sum(a(:,i)==ii);    %   confusion matrix
   end
   per_class_classification_rate(i)=I(i,i)/Test_Class_Size;
end
```

- FVS-LDA_IRdata_Identification.m

```
%                                                                              %
********************************************************************************
% % File name:       FVS-LDA_IRdata_Identification.m
% % Thesis Advisor: Prof.M.P.Fargues Naval Postgraduate School
% % Author:         Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
% % Date:           March 2006
% % Description:    Implements the FVS-LDA scheme on IR data using the
% %                 global parameters obtained via cross-validation. The
% %                 identification type of experiment is considered for
% %                 performance evaluation. The associated confidence
% %                 intervals for a 95% significance level are estimated.
% % Parameters      1.Kernel mapping function
% %                 2.Kernel global parameters
% %                 3.Number of FVs
% %                 4.Distance simmilarity metrics(i.e. euclidean norm,
% %                   euclidean angle, mahalanobis angle)

% % Outputs         1.CMC plots
% %                 2.Rank one performance metrics
% %                 3.Confidence intervals
% %                 4.Elapsed time
% % Functions used RankScore.m, confint.m, myseparation.m
```

```
%
%*************************************************************************
**


clc;
clear;
load    A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A);
clear  A A_all;

% %CONSTANTS DECLARATION

Train_Class_Size= 18;                  % Number of Training Images per Class
Test_Class_Size = 12;                  % Number of Testing Images per Class
Num_Class = 50;                        % Total Number of Classes
Sample_Class=30;                       % Samples per class
Num_Train_Imag =Train_Class_Size*Num_Class; % Total Number of Training Images
Num_Test_Imag =Test_Class_Size*Num_Class;   % Total Number of Testing Images
Num_iter=1000;                         % Number of Iterations 1000 used
train=[];test=[];
Atemp=Atemp-mean(mean(Atemp));
tic
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for iterations=1:Num_iter          %ITERATIONS LOOP(OUTER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % DATA SEPARATION INTO TRAINING & TESTING
[A1,A2]=myseparation(Atemp,Sample_Class,Num_Class,Train_Class_Size);

% %FEATURE VECTORS SELECTION
iterations
fprintf('Feature Vector Selection in progress ...\n');            % set up the global parameter
p1= sigma^2 for a gaussian kernel

Mapping='gauss';
global p1;
p1=14*10^(6);                                          % k=exp(-dist2(x2,x1)/p1),sigma^2 =
p1,for gaussian mapping

% % Mapping='poly';
% % global p1;
```

```
% % p1=2;                                          % k=(x2*x1').^p1,order = p1 for
polynomial mapping

% % Mapping='poly2';
% % global p1;
% % p1=2;                                         % k=(x2*x1'+1).^p1,order = p1 for
polynomial mapping

% % Mapping='sigmoid';
% % global p1;
% % global p2;
% % p1=9*10^(-9);                                  % k=tanh(p1*x2*x1'+p2),sigmoid
mapping
% % p2=0;

Features=250;
Global_Fitness=1-eps;
dataFVS=FVSelection2(A1',Mapping,Features,Global_Fitness);          % Select feature
vectors among the training data
S=dataFVS.S;                                    %Features selected
Ptrain=dataFVS.P;                                %Training data projected onto the
feature vectors subspace
Ptest=FVSKernel2(Mapping,A2',A1(:,S)')';                %Projection of all the testing
data onto the subspace of F
                                        %the FVs space(explicit projection using the
kernels)

% %APPLY FLD
%FLD is applied in the projected data onto the FVs
%subspace once they are linearly separable


%Create C matrix
for i=1:Num_Class;
   c(:,i)=i*ones(1,Train_Class_Size);
end
C=reshape(c,1,Train_Class_Size*Num_Class);
[W,D]=fld(Ptrain',C);
X_train_proj=W'*Ptrain';          % Projected Training Data onto FLD space
X_test_proj=W'*Ptest';           % Projected Testing Data onto FLD space

% %DISCRIMINANT FUNCTION
Distance='euc_angular';
[per_class_classification_rate,I,g]=discriminate(X_train_proj,X_test_proj,Distance);
%discriminate returns the per class classification rate,
%the confusion matrix I
```

```
%the discriminant's values g
overall_classification_rate(iterations)=100*sum(diag(I(:,:)))/Num_Test_Imag;

% %RANK SCORE(CUMMULATIVE MATCHING SCORE)

[Rank]=RankScore(g',Num_Class,Test_Class_Size,Num_Test_Imag);
Rank_FVS(iterations,:) = Rank;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end   %ITERATIONS LOOP(OUTER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time=toc;
classification_rate=mean(overall_classification_rate)
% % CONFIDENCE INTERVAL

interval=confint(overall_classification_rate,0.05,iterations);

% Results returned in a structure


clear A1 col A2 dataFVS Atemp dim1 C dim2 D g I i Num_Class ii Num_Test_Imag
index Num_Train_Imag k Num_iter kk
clear Ptest Ptrain mu  S n Sample_Class  Test_Class_Size  iterations
clear Train_Class_Size row W  X_test_proj test X_train_proj train a value c  cl

Results=struct('Mapping',Mapping,'FVs',Features,'Order',p1,          'Performance',
classification_rate,'ElapsedTime',time,'Interval',interval,'Distance',Distance);

% save Trial_mapping_FVs

% figure(1)
% stem(mean(Rank_FVS));grid
% title('Classifier based on the FVS data preprocessing')
% ylabel('Cummulative Rank Score')
% xlabel('Rank Index')
% axis ([-2 52,0 1.2])
%
% figure(2)
% stem(overall_classification_rate)
% title(' FVS-LDA classifier ')
% ylabel('Classification rate')
% xlabel('Iteration Index')
```

- FVS_LDA_IRdata_Verification.m

```
%                                                                       %
************************************************************************
% % File name:       FVS_LDA_IRdata_Verification.m
% % Thesis Advisor: Prof.M.P.Fargues Naval Postgraduate School
% % Author:        Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
% % Date:          March 2006
% % Description:    Implements the FVS-LDA scheme on IR data using the
% %                 global parameters obtained via cross-validation. The
% %                 verification type of experiment is considered for
% %                 performance evaluation. The associated confidence
% %                 intervals for a 95% significance level are estimated.
% % Parameters      1.Kernel mapping function
% %                 2.Kernel global parameters
% %                 3.Number of FVs (max 540 for 30 classes used for training)
% %                 4.Distance similarity metrics(only
% %                   Euclidean angle, Mahalanobis angle that are evaluated
% %                   between 0 and pi and can be normalized)
% % Outputs          1.ROC plots (EERs estimated graphically from the ROCs)
% %                 2.Rank one performance metrics
% %                 3.Elapsed time
% % Functions used:  FVSelection2.m, testKFA2.m, FVSProjection2.m, fld.m
% %                 FVSKernel2.m, applyKFA2.m, dist.m, invrec.m, sortem.m
% %                 myseparation.m, RankScore.m, confint.m
%
%************************************************************************
**
clc;
clear;
load    A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A);
clear  A A_all;

% %CONSTANTS DECLARATION

Train_Class_Size= 18;                % Number of Training Images per Class
Test_Class_Size = 12;                % Number of Testing Images per Class
Num_Class = 30;                      % Total Number of Classes
Sample_Class=30;                     % Samples per class
Num_Train_Imag =Train_Class_Size*Num_Class; % Total Number of Training Images
Num_Test_Imag =Test_Class_Size*Num_Class;   % Total Number of Testing Images
Num_iter=100;                        % Number of Iterations 1000 used
train=[];test=[];
Atemp=Atemp-mean(mean(Atemp));
tic
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
for iterations=1:Num_iter          %ITERATIONS LOOP(OUTER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%

%select only 30 classes for your gallery of 18 samples per class for
%training

Agallery=Atemp(:,1:900);
[A1,A2]=myseparation(Agallery,Sample_Class,Num_Class,Train_Class_Size);
Aprobes=[A2 Atemp(:,901:1500)];

% %FEATURE VECTORS SELECTION
iterations
fprintf('Feature Vector Selection in progress ...\n');          % set up the global parameter
p1= sigma^2 for a gaussian kernel

% % Mapping='gauss';
% % global p1;
% % p1=9*10^(6);                                % k=exp(-dist2(x2,x1)/p1),sigma^2
= p1,for gaussian mapping
% %
Mapping='poly';
global p1;
p1=2;                                   % k=(x2*x1').^p1,order = p1 for polynomial
mapping

% % Mapping='poly2';
% % global p1;
% % p1=2;                                       % k=(x2*x1'+1).^p1,order = p1 for
polynomial mapping

% Mapping='sigmoid';
% global p1;
% global p2;
% p1=9*10^(-9);                                  % k=tanh(p1*x2*x1'+p2),sigmoid
mapping
% p2=0;

Features=150;
Global_Fitness=1-eps;
dataFVS=FVSelection2(A1',Mapping,Features,Global_Fitness);          % Select feature
vectors among the training data
S=dataFVS.S;                                  %Features selected
```

```
Ptrain=dataFVS.P;                                   %Training data projected onto the
feature vectors subspace
Ptest=FVSKernel2(Mapping,Aprobes',A1(:,S)')';                   %Projection of all the
testing data onto the subspace of F

% %APPLY FLD
%FLD is applied in the projected data onto the FVs
%subspace once they are linearly separable


%Create C matrix
for i=1:Num_Class;
   c(:,i)=i*ones(1,Train_Class_Size);
end
C=reshape(c,1,Train_Class_Size*Num_Class);
[W,D]=fld(Ptrain',C);
X_train_proj=W'*Ptrain';          % Projected Training Data onto FLD space
X_test_proj=W'*Ptest';           % Projected Testing Data onto FLD space

% %DISCRIMINANT FUNCTION
Train_Class_Size= 18;                % Number of Training Images per Class
Test_Class_Size = 12;                % Number of Testing Images per Class
Num_Class = 30;                     % Total Number of Classes

for i=1:Num_Class;
   cl=X_train_proj(:,(1+Train_Class_Size*(i-1):Train_Class_Size*i)); % class separation
   mu(:,i)=mean(cl');
   % NOTE!!! mu is the mean vector of each class or Class Centroids
   % is considered as the Gallery Set since it contains one
   % signature(i.e. class centroid) per class

end

%used for mahalanobis angular
matrix_1=cov(X_train_proj');
sigma=sqrt(diag(matrix_1));
matrix_2=diag((sigma).^-1);

[row,col]=size(X_test_proj);


for n=1:col;
   for i=1:Num_Class;
% % Distance='euc_angular';   % euclidean angular
% %          g(i,n)= acos( ( ( X_test_proj(:,n)'* mu(:,i) ) ) / ( norm( X_test_proj(:,n),2 ) *
norm( mu(:,i),2 ) ) );
```

```
        Distance='mah_angular' ;  % mahalanobis angular
        g(i,n)= acos( ( ( X_test_proj(:,n)' * matrix_2 ) * ( matrix_2 * mu(:,i) ) ) / ( norm(
X_test_proj(:,n).*((sigma).^-1)  ,2  )  *  norm(  mu(:,i).*((sigma).^-1)  ,2  )  ) );
% MAHALANOBIS ANGULAR DISTANCE

    end
end

% %RANK SCORE(CUMMULATIVE MATCHING SCORE)

[Rank]=RankScore(g',Num_Class,Test_Class_Size,Num_Test_Imag);
Rank_FVS(iterations,:) = Rank;


%normalise the threshold

g=g/pi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Num_Class_Gallery=Num_Class;                %same as those used for training
Num_Class_Probe_CI=Num_Class;               %same as those used for training
Num_Class_Probe_FA=50-Num_Class;            %not condidered for training
samples_CI=Num_Class_Probe_CI*Test_Class_Size;
samples_FA=Num_Class_Probe_FA*Sample_Class;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for i=1:100;
    threshold=0.01*i;
    gcompare=g<threshold;
    g_CI=gcompare(:,1:samples_CI);
    g_FA=gcompare(:,samples_CI+1:(samples_CI+samples_FA));

    for n=1:Num_Class
     a_CI(n,:)=g_CI(n,1+Test_Class_Size*(n-1):Test_Class_Size*n);
    end

    PCI(i)=sum(sum(a_CI))/samples_CI;

    sFA=sum(g_FA)>=1;
    PFA(i)=sum(sFA)/samples_FA;

end
P_ci(iterations,:)=PCI;
```

```
P_fa(iterations,:)=PFA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end   %ITERATIONS LOOP(OUTER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % CONFIDENCE INTERVAL

interval=confint(overall_classification_rate,0.05,iterations);

time=toc;
clear A1 col A2 dataFVS Atemp dim1 C dim2 D g I i Num_Class ii Num_Test_Imag
index Num_Train_Imag k Num_iter kk
clear Ptest Ptrain mu  S n Sample_Class  Test_Class_Size  iterations Global_Fitness PFA
ans matrix_2 threshold
clear  Train_Class_Size  row  W   X_test_proj  test  X_train_proj  train  a  value  c   cl
Num_Class_Gallery g_CI sFA
clear Agallery Num_Class_Probe_CI g_FA  samples_CI Aprobes Num_Class_Probe_FA
gcompare samples_FA  PCI a_CI matrix_1 sigma

classification_rate=mean(Rank_FVS(:,1))
Results=struct('Mapping',Mapping,'FVs',Features,'Order',p1,          'Performance',
classification_rate,'ElapsedTime',time,'Distance',Distance);
% save Trial_FVSLDA_roc_1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1)
subplot(2,1,1),plot(mean(P_fa),1-mean(P_ci),'-rs')
title('Verification Experimnt using ROC ')
xlabel('P_F_A Probability of False Alarm')
ylabel('1-P_C_I Probability of Incorrect Rejection')
axis([0 1.2 0 1.2])
line([1 -1],[1 -1])
grid
hold on
subplot(2,1,2),plot(mean(P_fa),mean(P_ci),'-gd')
hold off
title('Verification Experimnt using ROC ')
xlabel('P_F_A Probability of False Alarm')
ylabel('P_C_I Probability of Correct Identification')
axis([0 1.2 0 1.2])
line([1 -1],[1 -1])
grid
```

- GDA_IRdata_Verification.m

```
%% ********************************************************************
%% File name:        GDA_IRdata_Verification.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:           Captain Dimitrios I. Domboulas
%%                   Hellenic Air Force. Modified by
%%                   Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
%% Date:             May    2006

%% Description:      Performs the GDA method, for various kernel functions
%%                   and distances on IR data.The verification type of
%%                   experiment is considered for performance evaluation.
%%
%%
%% Inputs:           1. Kernel function type (Polynomal considered only)
%%                   2. Number of K matrix eigenvectors kept
%%                   3. Distance type to be used
%%                   4. Number of iterations
%% Outputs:          1. ROC plots(EERs estimated graphically from the ROCs)
%%                   2. Rank one performance metrics based on 30 classes
%%                   3. Classification Performance per Class
%%                   4. Elapsed Time
%% Functions used:   buildGDA_Opt.m,  eigensystem.m,  dataST.m,
%%                   spreadGDA_Opt.m, KernelFunction.m, myseparation.m
%% ********************************************************************

clc;
clear;
load    A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A);
clear  A A_all;

% %CONSTANTS DECLARATION

Train_Class_Size= 18;                  % Number of Training Images per Class
Test_Class_Size = 12;                  % Number of Testing Images per Class
Num_Class = 30;%%Modified by Alexandropoulos% Total Number of Classes
Sample_Class=30;                       % Samples per class
Num_Train_Imag =Train_Class_Size*Num_Class; % Total Number of Training Images
Num_Test_Imag =Test_Class_Size*Num_Class;   % Total Number of Testing Images
Num_iter=100;                          % Number of Iterations 1000 used
train=[];test=[];
```

```
Atemp=Atemp-mean(mean(Atemp));
cs = [18,18,18,18,18,18,18,18,18,18];
% Class_Sizes = [cs, cs, cs, cs, cs]; %%(1 X 50) vector, with class sizes per class
Class_Sizes = [cs, cs, cs]; %%(1 X 30) vector, with class sizes per class%%%%ROC

degree = 2;  %%% select Polynomial kernel degree, USE 0 ==> for GAUSSIAN kernel
ONLY
Num_ev = 150; %%% input Number of K matrix eigenvectors  (0 => Default)
select_dist = 5;  %%%  select distance
tic
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for iterations=1:Num_iter          %ITERATIONS LOOP(OUTER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
loop=iterations
%select only 30 classes for your gallery of 18 samples per class for
%training

Agallery=Atemp(:,1:900);%%Added by Alexandropoulos
[A1,A2]=myseparation(Agallery,Sample_Class,Num_Class,Train_Class_Size);%%Adde
d by Alexandropoulos
% Aprobes=[A2 Atemp(:,901:1500)];%%Added by Alexandropoulos

A1 = DataSt( A1 ); %% Mean =0, Standard Deviation = 1
A2 = DataSt( A2 ); %% Mean =0, Standard Deviation = 1

Le = A1'; %%% Learning Data (540 Images are rows)
Te_CI = A2'; %% 360 Testing images of our Data Base (Images are rows)
Te_FA =Atemp(:,901:1500)';%%Modified by Alexandropoulos %Imposter set
[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev );
%%run GDA
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA, degree ); %% Projected Training images
in GDA
Pgda_B_CI = spreadGDA_Opt( Te_CI, Le, dataGDA, degree ); %% Projected
Testing_CI images in GDA
Pgda_B_FA = spreadGDA_Opt( Te_FA, Le, dataGDA, degree ); %% Projected
Testing_FA images in GDA
Pgda_B = [Pgda_B_CI;Pgda_B_FA];%% Projected Testing_Total images in GDA

%%% FIND VARIANCE of each column of the 900X49 Projected Train Data Matrix.
%%% Covariance is used for other Image Distance types
for k = 1 : size( Pgda_A , 2 );
Pr_Tr_Dat_Cov_Col(k) = cov( Pgda_A(:,k) );  %% Row Vector 1X49 with variances
%%%  of each of the 49 columns of the Projected Training Data Matrix 900X49
end;
```

```
Pr_Tr_Dat_Cov_Mat = cov(Pgda_A); %% 49X49 covariance matrix of Proj Train Data
Matrix
%%%%%% It is Diagonal!!Invertible & Each Dimension 's features are
%%%%%% independent from each othert ==> It is the Ideal case
Inv_Cov = inv( Pr_Tr_Dat_Cov_Mat );  %%% Inverse Covar Matr
%%% For the Approximated Mahalanobis Distance
for k = 1 : size( Inv_Cov,2 );
   s_inv(k) = Inv_Cov(k,k); %% ROW VECTOR 1 x 49
end;
sr_s_inv = sqrt(s_inv(k));  %% Row Vector
sr_inv_diag = diag( sr_s_inv ); %%Diagonal Matrix with diag elem the sq roots of
%%% the inverse diag elem of cov matrix  49 x 49
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END                VARIANCE
SECTION

%% Find Centroids per Each Class Projected Training Images
for k= 1 : Num_Class
ProjTrainCentr( k , : ) = mean( Pgda_A( ((k-1)*Train_Class_Size + 1) : ( k *
Train_Class_Size ) , : ) );
end;

%%% create cell array
DistName = {'Norm-2            ';'Mahalanobis       ';
   'Mahalanobis Norm-2 ';'Mahalanobis Norm-1 ';'Mahalanobis Angular'};

%%% Find Various Distances for Testing Images
[a1 a2]=size(Pgda_B);
for k = 1 : a1;
   for m = 1: Num_Class;
      if select_dist == 1;    %% NORM-2 Distance
   TestDist(k,m) = norm( Pgda_B(k,:) - ProjTrainCentr(m,:),2  );

   elseif select_dist == 2;  %% MAHALANOBIS DISTANCE
   TestDist(k,m) = sqrt( (Pgda_B(k,:) - ProjTrainCentr(m,:)) * Inv_Cov * (Pgda_B(k,:) -
ProjTrainCentr(m,:))' ); %% Mahalanobis Distance

   elseif select_dist == 3; %% Mahalanobis NORM-2 Distance
   temp = ( sr_s_inv .* ( Pgda_B(k,:) - ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_B(k,:) - ProjTrainCentr(m,:) ) )' ;
   TestDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

   elseif select_dist == 4; %% Mahalanobis NORM-1 Distance
   TestDist(k,m)=sr_s_inv .* norm( Pgda_B(k,:) - ProjTrainCentr(m,:),1 );

   else  %%%% MAHALANOBIS ANGULAR DISTANCE
```

```
   TestDist(k,m) = acos( ( ( Pgda_B(k,:) * sr_inv_diag ) * ( sr_inv_diag * ProjTrainCentr(m,:)' ) ) / ( norm( Pgda_B(k,:).*sr_s_inv ,2 ) * norm( ProjTrainCentr(m,:).*sr_s_inv ,2 ) ) );

   end;  %%% end if

  end
end
%%% Assign 1-50 Class Indeces for each of the 600 test Images
% T_B = [1,1,1,1,1,1,1,1,1,1,1,1];
T_B = ones(1,Test_Class_Size);
for m = 1 : Num_Class;
T_B( ( Test_Class_Size * (m-1) + 1 ) : ( Test_Class_Size * m )  ) = m * T_B( 1:Test_Class_Size) ;
end; %%% T_B contains the class number of each testing image (1X600)

%%% Sort Distances per Testing  Image - RANK
for k = 1 : Num_Test_Imag ;
temp1 = sort( TestDist(k,:) );
I(k) = find(  (temp1 == TestDist(k,T_B(k))*ones(1,Num_Class)) );
end;

%%% Performance Measurement Per Class
for  k = 1 : Num_Class;
   temp2 = find( I( ((k-1)*Test_Class_Size + 1) : ( k * Test_Class_Size ) ) == ones( 1,Test_Class_Size ) );
   Perf_Class(k) = size( temp2 , 2 ) / Test_Class_Size;
end;

%%% Find the probability of each of the 50 Ranks assigned to all the testing images
for k =1 : Num_Class ;
   temp3 = find( I == k * ones( 1,Num_Test_Imag ) );
   Prob(k) = size( temp3 , 2 ) / Num_Test_Imag;
end;

%%%%  RANK SCORE Evaluation - Cummulative Probability of Ranks - Incresing Order
Rank(1) = Prob(1);
for k = 2 : Num_Class ;
   Rank( k ) = Rank( k-1 ) + Prob( k );  %% Rank is 1X50 with accumulated Probability
end;

Rank_718(loop,:) = Rank;
Perf_Class_718(loop,:) = Perf_Class;

clear  A  TA  B  TB  f  L1  L2  Le  Te  dataGDA  centeredkM  kM;
```

```
clear Pgda_A Pgda_B k j Pr_Tr_Dat_Cov_Col Pr_Tr_Dat_Cov_Mat;
clear TrainDist temp Pgda_Aindex Class_Index;
clear Inv_Cov s_inv sr_inv_diag ProjTrainCentr DistName;
clear Pgda_BIndex Class_Index T_B temp1 I temp2 temp3;
clear Prob Delete d dim1 dim2 i m n sr_s_inv;
clear Rank Perf_Class Pgda_AIndex cs rp;


% % save Mah_Ang_Rank_PerfClas_GDA_0_7_040926 Rank_718 Perf_Class_718 ;
% % save Mah_Ang_Rank_PerfClas_GDA_2_100EV_041118 Rank_718
Perf_Class_718 ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Lines 174 to the end Added by Alexandropoulos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%normalize the threshold
g=TestDist';
g=g/pi;
Sample_Class=30;                    % Samples per class

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Num_Class_Gallery=Num_Class;            %same as those used for training
Num_Class_Probe_CI=Num_Class;           %same as those used for training
Num_Class_Imposter_FA=50-Num_Class;     %not condidered for training
samples_CI=Num_Class_Probe_CI*Test_Class_Size;
samples_FA=Num_Class_Imposter_FA*Sample_Class;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:100;
   threshold=0.01*i;
   gcompare=g<threshold;
   g_CI=gcompare(:,1:samples_CI);
   g_FA=gcompare(:,samples_CI+1:(samples_CI+samples_FA));

   for n=1:Num_Class
    a_CI(n,:)=g_CI(n,1+Test_Class_Size*(n-1):Test_Class_Size*n);
   end

   PCI(i)=sum(sum(a_CI))/samples_CI;

   sFA=sum(g_FA)>=1;
```

```
    PFA(i)=sum(sFA)/samples_FA;

end
P_ci(iterations,:)=PCI;
P_fa(iterations,:)=PFA;
end;        %%%%%%        END     OF     MAIN     ITERATION     LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time= toc; %% Stop timing each Main Iteration Loop

classification_rate=mean(Rank_718(:,1))
overall_classification_rate=Rank_718(:,1);
Mapping='polynomial';   %kernel function used
Distance='mah_angular';
% interval=confint(overall_classification_rate,0.05,iterations);

% % Results_GDAroc=struct('Mapping',Mapping,'Eigenvalues',Num_ev,'Order',degree,
'Performance', classification_rate,'ElapsedTime',time,'Distance',Distance);
% %
% % clear A1 g_CI A2 Perf_Class_718 g_FA  Agallery Pgda_B_CI  gcompare  Aprobes
Pgda_B_FA i  Atemp Rank_718 iterations
% % clear Class_Sizes Sample_Class loop Num_Class Te_CI n Num_Class_Gallery
Te_FA  Num_Class_Probe_CI sFA Num_Class_Probe_FA Test_Class_Size samples_CI
% % clear Num_Test_Imag Train_Class_Size samples_FA Num_Train_Imag a1
select_dist  a2 test Num_iter a_CI threshold  PCI PFA train
% %
% % save Trial_GDAroc_1


figure(1)
subplot(2,1,1),plot(mean(P_fa),1-mean(P_ci),'-rs')
title('Verification Experimnt using ROC ')
xlabel('P_F_A Probability of False Alarm')
ylabel('1-P_C_I Probability of Incorrect Rejection')
axis([0 1.2 0 1.2])
line([1 -1],[1 -1])
grid
hold on
subplot(2,1,2),plot(mean(P_fa),mean(P_ci),'-gd')
hold off
title('Verification Experimnt using ROC ')
xlabel('P_F_A Probability of False Alarm')
ylabel('P_C_I Probability of Correct Identification')
axis([0 1.2 0 1.2])
line([1 -1],[1 -1])
grid
```

- the5x2cvpaired_ttest.m

```
%                                                                       %
************************************************************************
% % File name:       the5x2cvpaired_ttest.m
% % Thesis Advisor: Prof.M.P.Fargues Naval Postgraduate School
% % Author:         Ioannis M.Alexandropoulos LTJG(E) Hellenic Navy
% % Date:           July 2006
% % Description:    Implements the the 5x2cv paired t test to compare
% %                 FVS-LDA versus GDA algorithm based on the estimated
% %                 classification rates for various number of feature
% %                 vectors/eigen vectors selected respectively.
% % Parameters      1.Kernel mapping function
% %                 2.Kernel global parameters
% %                 3.Number of FVs/eigenevectors
% %                 4.Distance similarity metrics
% % Outputs         1.Estimated value for the random variable following a t
% %                 distribution.
% %                 2.Rank one performance metrics
% %
% % Functions used:  FVSelection2.m,  FVSProjection2.m, fld.m
% %                 FVSKernel2.m,  dist.m, invrec.m, sortem.m
% %                 myseparation.m, RankScore.m, confint.m
% %                 buildGDA_Opt.m,  eigensystem.m,  dataST.m,
% %                 spreadGDA_Opt.m, KernelFunction.m, myseparation.m
%
%************************************************************************
**

clc;
clear;
load    A_all;
clear  img  Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w i;
Atemp = double(A);
clear  A A_all;

% %CONSTANTS DECLARATION

Train_Class_Size= 15;                % Number of Training Images per Class
Test_Class_Size = 15;                % Number of Testing Images per Class
Num_Class = 50;                      % Total Number of Classes
Sample_Class=30;                     % Samples per class
Num_Train_Imag =Train_Class_Size*Num_Class; % Total Number of Training Images
Num_Test_Imag =Test_Class_Size*Num_Class;   % Total Number of Testing Images
Num_replic=5;                        % Number of Iterations 1000 used
```

```
train=[];test=[];
Atemp=Atemp-mean(mean(Atemp));
for step=1:12;
   step
Num_ev = 50*step; %%% input Number of K matrix eigenvectors  (0 => Default)
Features=50*step;

for replications=1:Num_replic;
   replications
[A1,A2]=myseparation(Atemp,Sample_Class,Num_Class,Train_Class_Size);

cs = [15,15,15,15,15,15,15,15,15,15];
Class_Sizes = [cs, cs, cs, cs, cs]; %%(1 X 50) vector, with class sizes per class
degree = 2;  %%% select Polynomial kernel degree, USE 0 ==> for GAUSSIAN kernel
ONLY
% Num_ev = 100; %%% input Number of K matrix eigenvectors  (0 => Default)
select_dist = 5;  %%%  select distance


A1 = DataSt( A1 ); %% Mean =0, Standard Deviation = 1
A2 = DataSt( A2 ); %% Mean =0, Standard Deviation = 1
Le = A1'; %%% Learning Data (900 Images are rows)
Te = A2'; %%%  Testing Data (600 Images are rows)

[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev );
%%run GDA
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA, degree ); %% Projected Training images
in GDA
Pgda_B = spreadGDA_Opt( Te, Le, dataGDA, degree ); %% Projected Testing images in
GDA

%%% FIND VARIANCE of each column of the 900X49 Projected Train Data Matrix.
%%% Covariance is used for other Image Distance types
for k = 1 : size( Pgda_A , 2 );
Pr_Tr_Dat_Cov_Col(k) = cov( Pgda_A(:,k) );  %% Row Vector 1X49 with variances
%%%  of each of the 49 columns of the Projected Training Data Matrix 900X49
end;
Pr_Tr_Dat_Cov_Mat = cov(Pgda_A); %% 49X49 covariance matrix of Proj Train Data
Matrix
%%%%%%% It is Diagonal!!Invertible & Each Dimension 's features are
%%%%%%% independent from each othert ==> It is the Ideal case
Inv_Cov = inv( Pr_Tr_Dat_Cov_Mat );  %%% Inverse Covar Matr
%%% For the Approximated Mahalanobis Distance
for k = 1 : size( Inv_Cov,2 );
   s_inv(k) = Inv_Cov(k,k); %% ROW VECTOR 1 x 49
end;
```

```
sr_s_inv = sqrt(s_inv(k));  %% Row Vector
sr_inv_diag = diag( sr_s_inv ); %%Diagonal Matrix with diag elem the sq roots of
%%% the inverse diag elem of cov matrix  49 x 49
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END                VARIANCE
SECTION

%% Find Centroids per Each Class Projected Training Images
for k= 1 : Num_Class
ProjTrainCentr( k , : ) = mean( Pgda_A( ((k-1)*Train_Class_Size + 1) : ( k *
Train_Class_Size ) , : ) );
end;

%%% create cell array
DistName = {'Norm-2           ';'Mahalanobis       ';
   'Mahalanobis Norm-2 ';'Mahalanobis Norm-1 ';'Mahalanobis Angular'};

%%% Find Various Distances for Testing Images
for k = 1 : Num_Test_Imag;
   for m = 1: Num_Class;
       if select_dist == 1;    %% NORM-2 Distance
   TestDist(k,m) = norm( Pgda_B(k,:) - ProjTrainCentr(m,:),2  );

   elseif select_dist == 2;  %% MAHALANOBIS DISTANCE
   TestDist(k,m) = sqrt( (Pgda_B(k,:) - ProjTrainCentr(m,:)) * Inv_Cov * (Pgda_B(k,:) -
ProjTrainCentr(m,:))' ); %% Mahalanobis Distance

   elseif select_dist == 3; %% Mahalanobis NORM-2 Distance
   temp = ( sr_s_inv .* ( Pgda_B(k,:) - ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_B(k,:) - ProjTrainCentr(m,:) ) )' ;
   TestDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

   elseif select_dist == 4; %% Mahalanobis NORM-1 Distance
   TestDist(k,m)=sr_s_inv .* norm( Pgda_B(k,:) - ProjTrainCentr(m,:),1 );

   else  %%%% MAHALANOBIS ANGULAR DISTANCE
   TestDist(k,m) = acos( ( ( Pgda_B(k,:) * sr_inv_diag ) * ( sr_inv_diag *
ProjTrainCentr(m,:)' ) ) / ( norm( Pgda_B(k,:).*sr_s_inv ,2 ) * norm(
ProjTrainCentr(m,:).*sr_s_inv ,2 ) ) );

   end;  %%% end if

  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%GDA CONFUSION MATRIX%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
g=TestDist';
[row,col]=size(g)

for n=1:col;
[value(n),index(n)]=min(g(:,n)); % find where min occurs
end

% plot(index);

% %CONFUSION MATRIX

for i=1:Num_Class
    a(:,i)=index(1+Test_Class_Size*(i-1):Test_Class_Size*i);
    for ii=1:Num_Class
    I(ii,i)=sum(a(:,i)==ii);    %   confusion matrix
    end
    per_class_classification_rate(i)=I(i,i)/Test_Class_Size;
end
%GDA diagonal elements matrix check for correct classification
gg_GDA=zeros(size(g));

for n=1:col;
    gg_GDA(:,n)=g(:,n)<=min(g(:,n));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FVS
LDA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %FEATURE VECTORS SELECTION

fprintf('Feature Vector Selection in progress ...\n');              % set up the global parameter
p1= sigma^2 for a gaussian kernel

% % Mapping='gauss';
% % global p1;
% % % p1=9*10^(6);                                                        % k=exp(-
dist2(x2,x1)/p1),sigma^2 = p1,for gaussian mapping
% % p1=14*10^(6);                                       % k=exp(-dist2(x2,x1)/p1),sigma^2
= p1,for gaussian mapping

Mapping='poly2';
global p1;
```

```
p1=2;                                                % k=(x2*x1'+1).^p1,order = p1 for
polynomial mapping

% Mapping='sigmoid';
% global p1;
% global p2;
% p1=9*10^(-9);                                      % k=tanh(p1*x2*x1'+p2),sigmoid
mapping
% p2=0;

% Features=100;
Global_Fitness=1-eps;
dataFVS=FVSelection2(A1',Mapping,Features,Global_Fitness);          % Select feature
vectors among the training data
S=dataFVS.S;                                       %Features selected
Ptrain=dataFVS.P;                                  %Training data projected onto the
feature vectors subspace
Ptest=FVSKernel2(Mapping,A2',A1(:,S)')';                    %Projection of all the testing
data onto the subspace of F
                                        %the FVs space(explicit projection using the
kernels)

% %APPLY FLD
%FLD is applied in the projected data onto the FVs
%subspace once they are linearly separable


%Create C matrix
for i=1:Num_Class;
   c(:,i)=i*ones(1,Train_Class_Size);
end
C=reshape(c,1,Train_Class_Size*Num_Class);
[W,D]=fld(Ptrain',C);
X_train_proj=W'*Ptrain';          % Projected Training Data onto FLD space
X_test_proj=W'*Ptest';           % Projected Testing Data onto FLD space

% %DISCRIMINANT FUNCTION
Distance='euc_angular';
[per_class_classification_rate,I,g]=discriminate(X_train_proj,X_test_proj,Distance);

%FVS diagonal elements matrix check for correct classification
 gg_FVS=zeros(size(g));

for n=1:col;
   gg_FVS(:,n)=g(:,n)<=min(g(:,n));
end
```

```
for a=1:Num_Class;
   a_g(a,:)=gg_GDA(a,Test_Class_Size*(a-1)+1:Test_Class_Size*a);%GDA      diagonal
elements matrix check for correct classification

   a_f(a,:)=gg_FVS(a,Test_Class_Size*(a-1)+1:Test_Class_Size*a);%FVS      diagonal
elements matrix check for correct classification

end

% McNemar 2by2 matrix
% recall and,or,xor tables

%%%n00
%number of examples misclassified by both algorithms
n00=Test_Class_Size*Num_Class-sum(sum(or(a_g,a_f)));

%%%n11
%number of examples misclassified by neither algorithm
n11=sum(sum(and(a_g,a_f)));

%%%n01
%number of examples misclassified by GDA but not by FVS
n01=sum(sum(not(a_g)))-n00;

%%%n10
%number of examples misclassified by FVS but not by GDA
n10=Test_Class_Size*Num_Class-n00-n01-n11;

% McNemar 2by2 matrix
Table1(:,:,replications)=[n00+n01;n10+n11];

disp('Error Estimates')
disp('GDA')
pA1=(n00+n01)/(n00+n01+n10+n11)%classification error rate for GDA

disp('FVS')
pB1=(n00+n10)/(n00+n01+n10+n11)%classification error rate for FVS

clear centeredkM col dataFVS D dataGDA DistName Distance  Global_Fitness g
clear I gg_FVS Inv_Cov gg_GDA Le i Mapping ii index k kM m Pgda_A n Pgda_B n00
clear Pr_Tr_Dat_Cov_Col n01 Pr_Tr_Dat_Cov_Mat n10 ProjTrainCentr n11 Ptest Ptrain
S
clear per_class_classification_rate Te row TestDist s_inv sr_inv_diag
clear W sr_s_inv X_test_proj X_train_proj a value a_f a_g
```

```
%%%%%%%%%%%%%SWITCH TRANING AND TESTING PORTIONS%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Le = A2'; %%% Learning Data (900 Images are rows)
Te = A1'; %%%  Testing Data (600 Images are rows)

[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev ); %%run GDA
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA, degree ); %% Projected Training images in GDA
Pgda_B = spreadGDA_Opt( Te, Le, dataGDA, degree ); %% Projected Testing images in GDA

%%% FIND VARIANCE of each column of the 900X49 Projected Train Data Matrix.
%%% Covariance is used for other Image Distance types
for k = 1 : size( Pgda_A , 2 );
Pr_Tr_Dat_Cov_Col(k) = cov( Pgda_A(:,k) );  %% Row Vector 1X49 with variances
%%%  of each of the 49 columns of the Projected Training Data Matrix 900X49
end;
Pr_Tr_Dat_Cov_Mat = cov(Pgda_A); %% 49X49 covariance matrix of Proj Train Data Matrix
%%%%%% It is Diagonal!!Invertible & Each Dimension 's features are
%%%%%% independent from each othert ==> It is the Ideal case
Inv_Cov = inv( Pr_Tr_Dat_Cov_Mat );  %%% Inverse Covar Matr
%%% For the Approximated Mahalanobis Distance
for k = 1 : size( Inv_Cov,2 );
   s_inv(k) = Inv_Cov(k,k); %% ROW VECTOR 1 x 49
end;
sr_s_inv = sqrt(s_inv(k));  %% Row Vector
sr_inv_diag = diag( sr_s_inv ); %%Diagonal Matrix with diag elem the sq roots of
%%% the inverse diag elem of cov matrix  49 x 49
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END VARIANCE SECTION

%% Find Centroids per Each Class Projected Training Images
for k= 1 : Num_Class
ProjTrainCentr( k , : ) = mean( Pgda_A( ((k-1)*Train_Class_Size + 1) : ( k * Train_Class_Size ) , : ) );
end;

%%% create cell array
DistName = {'Norm-2            ';'Mahalanobis       ';
```

```
   'Mahalanobis Norm-2 ';'Mahalanobis Norm-1 ';'Mahalanobis Angular'};

%%% Find Various Distances for Testing Images
for k = 1 : Num_Test_Imag;
   for m = 1: Num_Class;
       if select_dist == 1;    %% NORM-2 Distance
   TestDist(k,m) = norm( Pgda_B(k,:) - ProjTrainCentr(m,:),2  );

   elseif select_dist == 2;  %% MAHALANOBIS DISTANCE
   TestDist(k,m) = sqrt( (Pgda_B(k,:) - ProjTrainCentr(m,:)) * Inv_Cov * (Pgda_B(k,:) -
ProjTrainCentr(m,:))' ); %% Mahalanobis Distance

   elseif select_dist == 3; %% Mahalanobis NORM-2 Distance
   temp = ( sr_s_inv .* ( Pgda_B(k,:) - ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_B(k,:) - ProjTrainCentr(m,:) ) )' ;
   TestDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

   elseif select_dist == 4; %% Mahalanobis NORM-1 Distance
   TestDist(k,m)=sr_s_inv .* norm( Pgda_B(k,:) - ProjTrainCentr(m,:),1 );

   else  %%%% MAHALANOBIS ANGULAR DISTANCE
   TestDist(k,m) = acos( ( ( Pgda_B(k,:) * sr_inv_diag ) * ( sr_inv_diag *
ProjTrainCentr(m,:)' ) ) / ( norm( Pgda_B(k,:).*sr_s_inv ,2 ) * norm(
ProjTrainCentr(m,:).*sr_s_inv ,2 ) ) );

   end;  %%% end if

  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%GDA CONFUSION MATRIX%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g=TestDist';
[row,col]=size(g)

for n=1:col;
[value(n),index(n)]=min(g(:,n)); % find where min occurs
end

% plot(index);

% %CONFUSION MATRIX

for i=1:Num_Class
   a(:,i)=index(1+Test_Class_Size*(i-1):Test_Class_Size*i);
   for ii=1:Num_Class
```

```
   I(ii,i)=sum(a(:,i)==ii);    %   confusion matrix
   end
   per_class_classification_rate(i)=I(i,i)/Test_Class_Size;
end
%GDA diagonal elements matrix check for correct classification
gg_GDA=zeros(size(g));

for n=1:col;
   gg_GDA(:,n)=g(:,n)<=min(g(:,n));
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FVS
LDA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %FEATURE VECTORS SELECTION

fprintf('Feature Vector Selection in progress ...\n');            % set up the global parameter
p1= sigma^2 for a gaussian kernel

% % Mapping='gauss';
% % global p1;
% % % p1=9*10^(6);                                                  % k=exp(-
dist2(x2,x1)/p1),sigma^2 = p1,for gaussian mapping
% % p1=14*10^(6);                                  % k=exp(-dist2(x2,x1)/p1),sigma^2
= p1,for gaussian mapping

Mapping='poly2';
global p1;
p1=2;                                              % k=(x2*x1'+1).^p1,order = p1 for
polynomial mapping

% Mapping='sigmoid';
% global p1;
% global p2;
% p1=9*10^(-9);                                     % k=tanh(p1*x2*x1'+p2),sigmoid
mapping
% p2=0;

% Features=100;
Global_Fitness=1-eps;
```

```
dataFVS=FVSelection2(A2',Mapping,Features,Global_Fitness);          % Select feature vectors among the training data
S=dataFVS.S;                                    %Features selected
Ptrain=dataFVS.P;                                   %Training data projected onto the feature vectors subspace
Ptest=FVSKernel2(Mapping,A1',A2(:,S)')';                    %Projection of all the testing data onto the subspace of F
                                        %the FVs space(explicit projection using the kernels)

% %APPLY FLD
%FLD is applied in the projected data onto the FVs
%subspace once they are linearly separable


%Create C matrix
for i=1:Num_Class;
   c(:,i)=i*ones(1,Train_Class_Size);
end
C=reshape(c,1,Train_Class_Size*Num_Class);
[W,D]=fld(Ptrain',C);
X_train_proj=W'*Ptrain';          % Projected Training Data onto FLD space
X_test_proj=W'*Ptest';            % Projected Testing Data onto FLD space

% %DISCRIMINANT FUNCTION
Distance='euc_angular';
[per_class_classification_rate,I,g]=discriminate(X_train_proj,X_test_proj,Distance);

%FVS diagonal elements matrix check for correct classification
 gg_FVS=zeros(size(g));

for n=1:col;
   gg_FVS(:,n)=g(:,n)<=min(g(:,n));
end


for a=1:Num_Class;
   a_g(a,:)=gg_GDA(a,Test_Class_Size*(a-1)+1:Test_Class_Size*a);%GDA      diagonal elements matrix check for correct classification

   a_f(a,:)=gg_FVS(a,Test_Class_Size*(a-1)+1:Test_Class_Size*a);%FVS      diagonal elements matrix check for correct classification

end

% McNemar 2by2 matrix
```

```
% recall and,or,xor tables

%%%n00
%number of examples misclassified by both algorithms
n00=Test_Class_Size*Num_Class-sum(sum(or(a_g,a_f)));

%%%n11
%number of examples misclassified by neither algorithm
n11=sum(sum(and(a_g,a_f)));

%%%n01
%number of examples misclassified by GDA but not by FVS
n01=sum(sum(not(a_g)))-n00;

%%%n10
%number of examples misclassified by FVS but not by GDA
n10=Test_Class_Size*Num_Class-n00-n01-n11;

% McNemar 2by2 matrix
Table2(:,:,replications)=[n00+n01;n10+n11];

disp('Error Estimates')
disp('GDA')
pA2=(n00+n01)/(n00+n01+n10+n11)%classification error rate for GDA

disp('FVS')
pB2=(n00+n10)/(n00+n01+n10+n11)%classification error rate for FVS

disp('estimated differences')
pp1(:,replications)=pA1-pB1
pp2=pA2-pB2
pmean=(pp1(:,replications)+pp2)/2
disp('estimated variance')
s2(:,replications)=(pp1(:,replications)-pmean)^2+(pp2-pmean)^2

clear centeredkM col dataFVS D dataGDA DistName Distance  Global_Fitness g
clear I gg_FVS Inv_Cov gg_GDA Le i Mapping ii index k kM m Pgda_A n Pgda_B n00
clear Pr_Tr_Dat_Cov_Col n01 Pr_Tr_Dat_Cov_Mat n10 ProjTrainCentr n11 Ptest Ptrain
S
clear per_class_classification_rate Te row TestDist s_inv sr_inv_diag
clear W sr_s_inv X_test_proj X_train_proj a value a_f a_g

end

t_hat(:,step)=(pp1(:,1))/sqrt(.2*sum(s2))
end
```

```
clear A1 A2 Num_replic dim1 replications  Atemp  p1 select_dist Num_Class train
clear A Sample_Class dim2 C Class_Sizes Test_Class_Size  Features  Train_Class_Size
test
clear Num_Test_Imag c Num_Train_Imag  cs  Num_ev  degree  ans

step=(1:12)*50;
figure(1)
plot(step,t_hat)
title('The 5x2cv paired t test on FVS-LDA and GDA schemes')
xlabel('Features/Eigenvalues')
ylabel('t_e_s_t_i_m_a_t_e')
text(50,-.8,' For a 95% confidence interval ')
text(50,-1.2,'the Null hypothesis is  true when')
text(50,-1.6,'-2.0515 < t_e_s_t_i_m_a_t_e < 2.0515')
grid
% save 5x2cvtest
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Bahlmann, C., B. Haasdonk, and H. Burkhardt. "Online Handwriting Recognition with Support Vector Machines-a Kernel Approach." *Frontiers in Handwriting Recognition, 2002.Proceedings.Eighth International Workshop on* (2002): 49-54.

Baudat, G., and F. Anouar. "Feature Vector Selection and Projection using Kernels." *Neurocomputing* 55, no. 1 (2003): 21–38.

Baudat, G., and F. Anouar. "Generalized Discriminant Analysis using a Kernel Approach." *Neural Computation* 12, (2000): 2385-2404.

Baudat, G., and F. Anouar. Kernel-Based Methods and Function Approximation." *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on* 2, (2001): 1244-1249.

Biometrics Catalog, "Biometrics Testing and Statistics," [http://www.biometricscatalog.org/NSTCSubcommittee/Documents/Biometrics%20Testing%20and%20Statistics.pdf], last accessed August 2006.

Cosmology Berkeley University, "Light and Spectra:Background," [http://cosmology.berkeley.edu/Education/DEMOS/Desk_Top_Stars/LS_Background.html], last accessed August 2006.

Centeno, T. P., and N. D. Lawrence. "Optimising Kernel Parameters and Regularisation Coefficients for Non-Linear Discriminant Analysis." *Journal of Machine Learning Research* 7, (2006): 455-491.

Chen, L. F., H. Y. M. Liao, M. T. Ko, J. C. Lin, and G. J. Yu. "New LDA-Based Face Recognition System which can Solve the Small Sample Size Problem." *Pattern Recognition* 33, no. 10 (2000): 1713-1726.

Chen, X., P. Flynn, and K. Bowyer. "Visible-Light and Infrared Face Recognition." *Proceedings of the Workshop on Multimodal User Authentication* (2003): 48–55.

Chen, X., P. J. Flynn, and K. W. Bowyer. "IR and Visible Light Face Recognition." *Computer Vision and Image Understanding* 99, no. 3 (2005): 332-358.

D'Agostino, RB, and M. Stephens. *Goodness-of-Fit Techniques* CRC Press, 1986.

Dietterich, T. G. "Statistical Test for Comparing Supervised Classification Learning Algorithms," Neural Computation, 10(7), 1895-1924, (1998).

Domboulas, D. I. *Infrared Imaging Face Recognition using Nonlinear Kernel-Based Classifiers.* Electrical Engineer Thesis, Naval Postgrduate School, Monterey, CA, (2004).

Duda, R. O., P. E. Hart, and D. G. Stork. *Pattern Classification* Wiley New York, 2001.

FLIR Infrared Camera Systems, “Thermal Imaging,” [http://www.flir.uk.com/assets/thermography.htm], last accessed August 2006.

Gnanadesikan, R. *Methods for Statistical Data Analysis of Multivariate Observations* Wiley New York, 1977.

GraphPad Software, “Normality Tests FAQ,” [http://www.graphpad.com/FAQ/viewfaq.cfm?faq=959], last accessed June 2006.

Grother, P. J., R. J. Michaels, and P. J. Phillips. Face Recognition Vendor Test 2002 Performance Metrics Springer, 2003.

Hastie, T., R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* Springer, 2001.

Huang, J., PC Yuen, W. S. Chen, and JH Lai. “Kernel Subspace LDA with Optimized Kernel Parameters on Face Recognition.” *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on* (2004): 327-332.

Infrared Solutions Inc., “IR–160 thermal imager,” [http://www.infraredsolutions.com/images/downloads/ImagerDS.pdf], last accessed July 2006.

Joachims, T. “Machine Learning” Cornell University Department of Computer Science [http://www.cs.cornell.edu/courses/cs478/2006sp/lectures/8-svm_kernels.pdf], last accessed June 2006.

Joachims, T. “Transductive Inference for Text Classification using Support Vector Machines.” *Proceedings of the Sixteenth International Conference on Machine Learning* (1999): 200–209.

Joachims, T “Text Categorization with Support Vector Machines: Learning with Many Relevant Features.” *Proceedings of the 10th European Conference on Machine Learning* (1998): 137-142.

Kim, S. W., and B. J. Oommen. “On using Prototype Reduction Schemes to Optimize Kernel-Based Nonlinear Subspace Methods.” *Pattern Recognition* 37, no. 2 (2004): 227-239.

Kocsor, A., and L. Toth. “Kernel-Based Feature Extraction with a Speech Technology Application.” *Signal Processing, IEEE Transactions on [See also Acoustics, Speech, and Signal Processing, IEEE Transactions on]* 52, no. 8 (2004): 2250-2263.

Kong, S. G., J. Heo, B. R. Abidi, J. Paik, and M. A. Abidi. “Recent Advances in Visual and Infrared Face recognition—a Review.” *Computer Vision and Image Understanding* 97, no. 1 (2005): 103-135.

Lee, C. K. *Infrared Face Recognition,* MSEE Thesis, Naval Postgrduate School, Monterey, CA (2004).

Liu, C., and H. Wechsler. "A Unified Bayesian Framework for Face Recognition." *Image Processing, 1998.ICIP 98.Proceedings.1998 International Conference on* 1, (1998): 151-155.

Liu, Q., R. Huang, H. Lu, and S. Ma. "Kernel-Based Optimized Feature Vectors Selection and Discriminant Analysis for Face Recognition." *Proceedings of International Conference on Pattern Recognition, Canada: Quebec* (2002).

Liu, Q., H. Lu, and S. Ma. "Improving Kernel Fisher Discriminant Analysis for Face Recognition." *Circuits and Systems for Video Technology, IEEE Transactions on* 14, no. 1 (2004): 42-49.

Lu, J., KN Plataniotis, and AN Venetsanopoulos. "Face Recognition using Kernel Direct Discriminant Analysis Algorithms." *Neural Networks, IEEE Transactions on* 14, no. 1 (2003): 117-126.

Lu, J., KN Plataniotis, and AN Venetsanopoulos. "Face Recognition using Kernel Direct Discriminant Analysis Algorithms." *Neural Networks, IEEE Transactions on* 14, no. 1 (2003): 117-126.

Muller, K. R., S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. "An Introduction to Kernel-Based Learning Algorithms." *Neural Networks, IEEE Transactions on* 12, no. 2 (2001): 181-201.

NASA, "The Electromagnetic Spectrum," [http://imagers.gsfc.nasa.gov/ems/infrared.html], last accessed August 2006.

NAVSEA Newport, "Curse of Dimensionality," [http://www.npt.nuwc.navy.mil/Csf/curse.html], last accessed June 2006.

Pereira, D. *Face Recognition using Uncooled Infrared Imaging*, Electrical Engineer Thesis, Naval Postgrduate School, Monterey, CA (2002).

Pontil, M., and A. Verri. "Support Vector Machines for 3D Object Recognition." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20, no. 6 (1998): 637-646.

Prokoski, F. "History, Current Status, and Future of Infrared Identification." *Computer Vision Beyond the Visible Spectrum: Methods and Applications, 2000.Proceedings. IEEE Workshop on* (2000): 5-14.

Schölkopf, B., and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.

Socolinsky, D., L. Wolff, J. Neuheisel, and C. Eveland. “Illumination Invariant Face Recognition using Thermal Infrared Imagery.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Pages I* (2001): 527–534.

Socolinsky, D. A., A. Selinger, and J. D. Neuheisel. “Face Recognition with Visible and Thermal Infrared Imagery.” *Computer Vision and Image Understanding* 91, no. 1-2 (2003): 72-114.

StatsDirect Ltd, “Confidence Intervals,” [http://www.camcode.com/help/statsdirect.htm], last accessed July 2006.

Theodoridis, S., and K. Koutroumbas. *Pattern Recognition* Academic Press, 2003.

Thomaz, CE, and DF Gillies. “A Maximum Uncertainty LDA-Based Approach for Limited Sample Size Problems—With Application to Face Recognition.” *Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on* (2005): 89-96.

Trujillo-Ortiz, A. and R. Hernandez-Walls. (2003). DagosPtest: D'Agostino-Pearson's K2 test for assessing normality of data using skewness and kurtosis. A MATLAB file. [http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3954&objectType=FILE], last accessed November 2005.

Vaswani, N., and R. Chellappa. “Principal Component Null Space Analysis for image/video Classification.” *Submitted to IEEE Trans. on Image Processing* (2004).

Wang, X., and X. Tang. “Dual-Space Linear Discriminant Analysis for Face Recognition.” *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference* on 2: 564-569.

Wang, X., and X. Tang. “Random Sampling LDA for Face Recognition.” *Computer Vision and Pattern Recognition, 2004. CVPR 2004 .Proceedings of the 2004 IEEE Computer Society Conference* on 2: 259-265.

Xiong, H., M. N. Swamy, and M. O. Ahmad. “Optimizing the Kernel in the Empirical Feature Space.” *IEEE Trans. Neural Netw.* 16, no. 2 (Mar, 2005): 460-474.

Yu, H., and J. Yang. “A Direct LDA Algorithm for High-Dimensional Data with Application to Face Recognition.” *Pattern Recognition* 34, no. 10 (2001): 2067-2070.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Jeffrey B. Knorr
   Chairman, Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Monique P. Fargues
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor Roberto Cristi
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

6. Professor Carlos Borges
   Department of Mathematics
   Naval Postgraduate School
   Monterey, California

7. Ioannis M. Alexandropoulos
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California